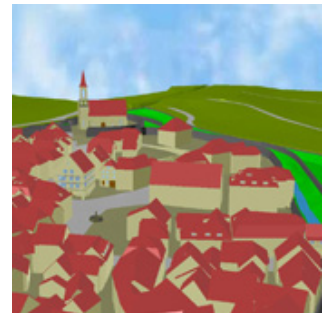
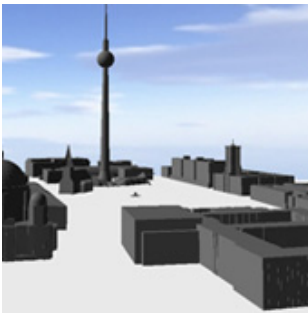


3D-Geo-Database for CityGML

Version 2.0.1

Documentation

April 24th 2009



**Institute for Geodesy and Geoinformation Science
Technische Universität Berlin**

Prof. Dr. Thomas H. Kolbe
Gerhard König
Claus Nagel
Alexandra Stadler

Content

DISCLAIMER.....	8
INTRODUCTION.....	9
1.1 RELATION TO PREVIOUS VERSION	9
1.2 FEATURES	10
1.3 SYSTEM AND DESIGN DECISIONS	12
2 DATA MODELLING	15
2.1 UML DIAGRAM OF THE 3D CITY MODEL	15
2.1.1 <i>Simplifications compared to CityGML 1.0.0 / 0.4.0</i>	15
2.1.1.1 Multiplicities, cardinalities and recursions	15
2.1.1.2 Data type adaptation	16
2.1.1.3 Project specific classes and class attributes	16
2.1.1.4 Simplified design of GML geometry classes	16
2.2 UML CLASS DIAGRAM	16
2.2.1 Spatial Model	17
2.2.1.1 Geometric-topological Model	17
2.2.1.2 Implicit Geometry	19
2.2.1.2 Appearance Model	19
2.2.1.3 Thematic model	22
2.2.1.3.1 Core Model	22
2.2.1.3.2 Building model	24
2.2.1.3.3 CityFurniture Model	27
2.2.1.3.4 Digital terrain model	28
2.2.1.3.5 Generic CityObject Model	29
2.2.1.3.6 LandUse Model	30
2.2.1.3.7 Transportation Model	31
2.2.1.3.8 Vegetation Model	33
2.2.1.3.9 WaterBodies Model	34
2.3 RELATIONAL DATABASE SCHEMA	36
2.3.1 <i>Mapping rules, schema conventions</i>	36
2.3.1.1 Mapping of classes onto tables	36
2.3.1.2 Explicit declaration of class affiliation	36
2.3.2 <i>Database schema</i>	36
2.3.2.1 Core Model	36
CITYOBJECT, CITYOBJECT_SEQ	36
CITYMODEL, CITYMODEL_SEQ	37
EXTERNAL_REFERENCE, EXTERNAL_REF_SEQ	37
CITYOBJECTGROUP, GROUP_TO_CITYOBJECT	37
2.3.2.2 Tables for geometry representation	39
SURFACE_GEOMETRY, SURFACE_GEOMETRY_SEQ	39
IS_XLINK	41
IS_REVERSE	42
2.3.2.3 Appearance Model	44
APPEARANCE, APPEARANCE_SEQ	44
SURFACE_DATA, APPEAR_TO_SURFACE_DATA	45
TEXTUREPARAM	45
OPENING	50
THEMATIC_SURFACE	50
BUILDING_INSTALLATION	52
ROOM	52
BUILDING_FURNITURE	52
ADDRESS, ADDRESS_TO_BUILDING, and ADDRESS_SEQ	52
2.3.2.4 CityFurniture Model	53
CITY_FURNITURE	53
2.3.2.5 Digital Terrain Model	54
RELIEF	54
2.3.2.6 Generic CityObject Model	55
GENERIC_CITYOBJECT	55
CITYOBJECT_GENERICATTRIB, CITYOBJECT_GENERICATT_SEQ	55
OBJECTCLASS	57
2.3.2.7 LandUse Model	58
LAND_USE	58

2.3.2.8	Orthophoto Model	59
	ORTHOPHOTO and ORTHOPHOTO_IMP	59
	ORTHOPHOTO_RDT	60
	ORTHOPHOTO_IMP_RDT	60
	ID sequences	60
2.3.2.9	Transportation Model	61
	TRAFFIC_AREA	61
	TRANSPORTATION_COMPLEX	61
2.3.2.10	Vegetation Model	62
	SOLITARY_VEGETAT_OBJECT	62
	PLANT_COVER	63
2.3.2.11	WaterBody Model	64
	WATERBODY, WATERBOD_TO_WATERBND_SRF	64
	WATERBOUNDARY_SURFACE	64
2.3.2.12	Sequences, Database_SRS	65
3	IMPLEMENTATION AND INSTALLATION	66
3.1	SYSTEM REQUIREMENTS	66
3.2	DATABASE SETUP	66
3.2.1	SQL scripts	66
3.2.2	Example session	68
4	CITYGML IMPORT / EXPORT TOOL	70
4.1	IMPORT / EXPORT TOOL INTERFACES	71
4.1.1	Command line interface	71
4.1.2	Graphical User interface	72
4.1.2.1	Database connection	72
4.1.2.2	CityGML File Import	73
4.1.2.3	CityGML Export	77
4.1.2.4	Preferences	79
4.1.2.4.1	Import preferences	79
4.1.2.4.2	Export preferences	90
4.1.2.4.3	General preferences	95
5	MATCHING TOOL	100
5.1	MOTIVATION	100
5.2	IDEA	100
5.3	APPROACH	100
5.4	GRAPHICAL USER INTERFACE	103
5.5	REALISATION: STORED PL/SQL PROCEDURES	106
5.5.1	Matching	106
5.5.1.1	create_matching_table	106
5.5.1.2	allocate_cand_building	107
5.5.1.3	allocate_geometry	107
5.5.1.4	rectify_geometry	107
5.5.1.5	aggregate_geometry	108
5.5.1.6	allocate_master_building	108
5.5.1.7	join_cand_master	109
5.5.1.8	function aggregate_mbr	109
5.5.1.9	aggregate_geometry_by_id	110
5.5.2	Merging	110
5.5.2.1	create_relevant_matches	110
5.5.2.2	collect_all_geometry	111
5.5.2.3	move_appearance	111
5.5.2.4	create_and_put_container	112
5.5.2.5	move_geometry	112
5.5.2.6	delete_multi_surfaces	112
5.5.2.7	cleanup	112
6	VERSION AND HISTORY MANAGEMENT	114
6.1	PREFACE	114
6.2	THE CONCEPT OF VERSIONS AND CITYMODELASPECTS	114
6.3	REALISATION IN ORACLE	117
6.3.1	AddPlanning	119
6.3.2	UpdatePlanning	119

6.3.3	<i>DiscardPlanning</i>	119
6.3.4	<i>AcceptPlanning</i>	120
6.3.5	<i>AddPlanningAlternative</i>	120
6.3.6	<i>UpdatePlanningAlternative</i>	120
6.3.7	<i>DiscardPlanningAlternative</i>	121
6.3.8	<i>GetDiff</i>	121
6.3.9	<i>GetAllDiff</i>	121
6.3.10	<i>GetConflicts</i>	121
6.3.11	<i>GetAllConflicts</i>	122
6.3.12	<i>RefreshPlanningAlternative</i>	122
6.3.13	<i>DeleteAllPlanningAlternatives</i>	122
6.3.14	<i>DeleteTermPlanningAlternatives</i>	123
6.3.15	<i>AddCityModelAspect</i>	123
6.3.16	<i>DeleteCityModelAspect</i>	123
6.3.17	<i>AddPAtoCMA</i>	123
6.3.18	<i>RemovePAfromCMA</i>	124
6.3.19	<i>DeleteAllCityModelAspects</i>	124
6.4	ADMINISTRATION PROGRAM "PLANNINGMANAGER"	124
6.4.1	<i>Plannings</i>	125
6.4.2	<i>Planning alternatives</i>	126
6.4.3	<i>Interface for the management of planning alternatives</i>	129
6.5	CONFLICT MANAGEMENT	129
6.5.1	<i>Differences</i>	129
6.5.2	<i>Conflicts</i>	129
6.6	USE IN APPLICATION PROGRAMS	130
7	TOOLS FOR RASTER DATA IMPORT AND EXPORT	132
7.1	SYSTEM REQUIREMENTS	132
7.2	USER INTERFACE	133
7.2.1	<i>The login dialog</i>	134
7.3	IMPORT	134
7.3.1	<i>Import of raster data</i>	134
7.3.1.1	Mosaicking of DTMs	135
7.3.1.2	Mosaicking of aerial images	135
7.4	EXPORT	136
7.4.1	<i>Export of raster data</i>	136
8	REFERENCES	138
9	APPENDIX A – SQL SCRIPTS	140
9.1	DATABASE	140
	CREATE_DB.sql	140
	CREATE_DB2.sql	142
	DATABASE_SRS.sql	144
	DO_NOTHING.sql	145
	HINT_ON_MISSING_SRS.sql	146
	OBJECTCLASS_INSTANCES.sql	147
	CITYOBJECT	150
	CITYOBJECT.sql	150
	CITYOBJECT_SEQ.sql	151
	CITYMODEL.sql	152
	CITYMODEL_SEQ.sql	153
	CITYOBJECT_MEMBER.sql	154
	EXTERNAL_REFERENCE.sql	155
	EXTERNAL_REF_SEQ.sql	156
	GENERALIZATION.sql	157
	IMPLICIT_GEOMETRY.sql	158
	IMPLICIT_GEOMETRY_SEQ.sql	159
	OBJECTCLASS.sql	160
	GEOMETRY	161
	SURFACE_GEOMETRY.sql	161
	SURFACE_GEOMETRY_SEQ.sql	162
	APPEARANCE	163
	APPEARANCE.sql	163
	APPEARANCE_SEQ.sql	164

	APPEAR_TO_SURFACE_DATA.sql.....	165
	SURFACE_DATA.sql.....	166
	SURFACE_DATA_SEQ.sql.....	167
	TEXTUREPARAM.sql.....	168
	BUILDING.....	169
	BUILDING.sql.....	169
	ADDRESS.sql.....	170
	ADDRESS_SEQ.sql.....	171
	ADDRESS_TO_BUILDING.sql.....	172
	BUILDING_FURNITURE.sql.....	173
	BUILDING_INSTALLATION.sql.....	174
	OPENING.sql.....	175
	OPENING_TO_THEM_SURFACE.sql.....	176
	ROOM.sql.....	177
	THEMATIC_SURFACE.sql.....	178
	CITYFURNITURE.....	179
	CITY_FURNITURE.sql.....	179
	CITYOBJECTGROUP.....	180
	CITYOBJECTGROUP.sql.....	180
	GROUP_TO_CITYOBJECT.sql.....	181
	DTM.....	182
	BREAKLINE_RELIEF.sql.....	182
	DTM_SEQ.sql.....	183
	MASSPOINT_RELIEF.sql.....	184
	RASTER_RELIEF.sql.....	185
	RASTER_RELIEF_IMP.sql.....	186
	RASTER_RELIEF_IMP_RDT.sql.....	187
	RASTER_RELIEF_RDT.sql.....	188
	RASTER_RELIEF_RDT_ID_TRIGGER.sql.....	189
	RASTER_RELIEF_RDT_IMP_ID_TRIGGER.sql.....	190
	RELIEF.sql.....	191
	RELIEF_COMPONENT.sql.....	192
	RELIEF_FEATURE.sql.....	193
	RELIEF_FEAT_TO_REL_COMP.sql.....	194
	TIN_RELIEF.sql.....	195
	GENERIC.....	196
	CITYOBJECT_GENERICATTRIB.sql.....	196
	CITYOBJECT_GENERICATT_SEQ.sql.....	197
	GENERIC_CITYOBJECT.sql.....	198
	LANDUSE.....	199
	LAND_USE.sql.....	199
	ORTHOPHOTO.....	200
	ORTHOPHOTO.sql.....	200
	ORTHOPHOTO_IMP.sql.....	201
	ORTHOPHOTO_RDT.sql.....	202
	ORTHOPHOTO_RDT_ID_TRIGGER.sql.....	203
	ORTHOPHOTO_RDT_IMP.sql.....	204
	ORTHOPHOTO_RDT_IMP_ID_TRIGGER.sql.....	205
	ORTHOPHOTO_SEQ.sql.....	206
	TRANSPORTATION.....	207
	TRAFFIC_AREA.sql.....	207
	TRANSPORTATION_COMPLEX.sql.....	208
	VEGETATION.....	209
	PLANT_COVER.sql.....	209
	SOLITARY_VEGETAT_OBJECT.sql.....	210
	WATERBODY.....	211
	WATERBODY.sql.....	211
	WATERBOUNDARY_SURFACE.sql.....	212
	WATERBOD_TO_WATERBND_SRF.sql.....	213
9.2	CONSTRAINTS.....	214
	ADD_CONSTRAINTS.sql.....	214
9.3	IMPORT PROCEDURES.....	222
	IMPORT_PROCEDURES.sql.....	222
	DUMMY_IMPORT.sql.....	223
9.4	RASTER MOSAIC.....	224
	MOSAIC.sql.....	224
9.5	TRIGGER.....	229
	TRIGGER.sql.....	229

9.6	INDEXES	230
	BUILD_SIMPLE_INDEX.sql	230
	SPATIAL_INDEX.sql	233
9.7	DATABASE VERSIONING	242
	ENABLEVERSIONING.sql	242
	DISABLEVERSIONING.sql	243
9.8	CREATE TABLES & PROCEDURES OF THE PLANNINGMANAGER	244
	CREATE_PLANNINGMANAGER.sql	244
	DROP_PLANNINGMANAGER.sql	245
9.9	SYSDBA	247
	SOLDNER_BERLIN_SRS_10G_R2.sql	247
9.10	UTILITIES	248
	GEODB_REPORT.sql	248
	DELETE_BUILDINGS.sql	251
9.11	DROP DATABASE	260
	DROP_DB.sql	260
9.12	PACKAGE GEODB	264
	CREATE_GEODB_PKG.sql	264
	DROP_GEODB_PKG.sql	265
	GEODB_IDX	266
	IDX.sql	266
	GEODB_STAT	271
	STAT.sql	271
	GEODB_UTIL	274
	UTIL.sql	274
9.13	MATCHING TOOL	277
	GEODB_MATCH	277
	MATCH.sql	277
	GEODB_PROCESS_MATCHES	283
	PROCESS_MATCHES.sql	283
	GEODB_DELETE_BY_LINEAGE	291
	DELETE_BY_LINEAGE.sql	291
10	APPENDIX B – DATABASE SETUP – EXAMPLE SESSION	302
11	APPENDIX C – LIST OF FIGURES AND TABLES	308

Disclaimer

3D City Database version 2.0 developed by the *Institute for Geodesy and Geoinformation Science (igg)* at the *Technical University Berlin* is free software under the GNU Lesser General Public License Version 3.0. See the file LICENSE shipped together with the 3D City Database software for more details. For a copy of the GNU Lesser General Public License see the files COPYING and COPYING.LESSER or visit <http://www.gnu.org/licenses/>.

THE SOFTWARE IS PROVIDED BY *igg* "AS IS" AND "WITH ALL FAULTS." *igg* 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.

igg 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 *igg*.

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

Introduction

In the period from November 2003 to December 2005 the official virtual 3D city model of Berlin, commissioned by The Berlin Senate and Berlin Partner GmbH, was developed within a pilot project funded by the European Union [Plümer et al. 2005, Berlin 3D].

The model plays a central role in the three-dimensional spatial data infrastructure of Berlin and opened up a multitude of applications for the public and private sector alike. As an example the virtual city model is successfully used for presentation of the business location, its urban development combined with application related information to politicians, investors, and the public in order to support civic participation, provide access to decision-making content, assist in policy-formulation, and control implementation processes.

Within the framework *Europäische Fonds für regionale Entwicklung* (EFRE II) the project *Geodatenmanagement in der Berliner Verwaltung – Amtliches 3D-Stadtmodell für Berlin* allowed for upgrading the official 3D city model based on the recent CityGML specification 0.4.0 in the year 2007. Major extensions to the Berlin model are related to indoor modelling (Level-of-Detail 4 – LoD4) and to the visualization of surface information varying in time and topic according to CityGML's appearance model. In August 2008, CityGML 1.0.0 became an adopted standard of the Open Geospatial Consortium (OGC).

This document describes the design and the components of the 3D geodatabase which has been developed and implemented by the *Institute for Geodesy und Geoinformation Science (IGG) of the Berlin University of Technology* on behalf of the *Berliner Senatsverwaltung für Wirtschaft, Arbeit und Frauen* and the *Berlin Partner GmbH* (former Wirtschaftsförderung Berlin International). The development is extending the work carried out at the *Institute for Cartography and Geoinformation (IKG) of the University of Bonn*. The relational database model was finally implemented and evaluated in cooperation with *3DGeo GmbH* (now Autodesk GmbH) in Potsdam.

Please note, that the 3D database described in this documentation and the related tools for import, export and matching can be used for arbitrary CityGML files of any city.

This document is based on the previous 3D-Geodatenbank Berlin V 1.0 documentation (in German language), the Candidate OpenGIS® CityGML Implementation Specification (City Geography Markup Language), Version 0.4.0 [Gröger et al. 2007], and the OpenGIS City Geography Markup Language (CityGML) Encoding Standard, Version 1.0.0 [Gröger et al. 2008].

1.1 Relation to previous version

As mentioned before, the main difference compared with the previous version of the 3D geodatabase [3D City Database] is the upgrade to the last versions of CityGML (0.4.0 and 1.0.0). In contrast to the previous release, the development now comprises a complete implementation of CityGML's classes, representing the most relevant 3D urban objects with respect to their geometrical, topological, semantical and appearance properties. A new import/export utility program allows for reading and writing of arbitrary sized CityGML models.

The most outstanding new features concern the introduction of LoD4 objects combined with a new appearance model. This results in following benefits:

The sophisticated building model allows a more detailed representation of exterior building elements and is completed by interior structures such as rooms, floors etc. Rooms are represented as volume or surface geometries and are semantically structured by inner wall

surfaces, top surfaces and floor surfaces. Surfaces of interior rooms can be furnished with different textures or materials. Furthermore, rooms may be equipped with a variety of furniture objects. These are prerequisites for a realistic navigation within the building.

In addition to semantics and geometry, features can have appearances, i.e. information about the observable properties of a feature's surface. In order to define a feature's appearances, the geometry model is extended allowing for the modelling of visual properties. Next to visual properties, the model comprises arbitrary surface-based themes, such as infrared radiation, noise pollution etc. Thus, it is a more comprehensive approach to represent information about a feature's surface.

1.2 Features

- **CityGML 0.4.0 and 1.0.0 compliant database:** The implementation defines the classes and relations for the most relevant topographic objects in cities and regional models with respect to their geometrical, topological, semantical, and appearance properties. Included are generalization hierarchies between thematic classes, aggregations, relations between objects, and spatial properties. This thematic information goes beyond graphic exchange formats and allows to employ virtual 3D city models for sophisticated analysis tasks in different application domains.
- **Support of four different kinds of multi-representations: Levels of detail, different appearances, planning versions and history:** Every geoobject as well as the DTM and aerial photographs can be represented in five different resolution or fidelity steps (Levels of Detail, LoD). With increasing LoD, objects do not only obtain a more precise and finer geometry, but also gain a thematic refinement.

Different appearance data may be stored for each city object. Appearance relates to any surface-based theme, e.g. infrared radiation or noise pollution, not just visual properties. Consequently, data provided by appearances can be used as input for both presentation and analysis of virtual 3D city models. The database supports feature appearances for an arbitrary number of themes per city model. Each LoD of a feature can have individual appearances. Appearances can represent – among others – textures and georeferenced textures. All texture images can be stored in the database.

The version and history management employs Oracle's Workspace Manager. It is largely transparent to application programs that work with the database. For administration of planning areas and embodied planning variations, the tool "PlanningManager" was implemented. Furthermore, procedures saved within the database (Stored Procedures) are provided, which allow the comfortable management of planning alternatives and versions via application programs. The work is based on previous developments at the University of Bonn.

- **Complex digital terrain models:** DTMs may be represented in four different ways in the 3D geodatabase: regular grids, triangular irregular networks (TINs), 3D mass points and 3D break lines. For every level of detail a complex DTM consisting of any number of DTM-components and DTM-types can be defined. Besides, it is possible to combine certain kinds of DTM representations for the same geographic area with each other (e.g. mass points and break lines or grids and break lines). Grid-based DTMs may be of arbitrary size and are composed from separate tiles to a single overall grid using the Oracle 10G R2 GeoRaster functionality.
- **Management of large aerial photographs:** Aerial photographs of any size may be stored and administered. With the help of the Oracle 10G R2 GeoRaster functionality

tiled, homogeneous photographs stored in the database can be aggregated to a single overall image and regions can be displayed seamlessly.

- **Complex building modelling:** The representation of buildings in the 3D geodatabase ranges from coarse models to geometrically and semantically fine grained structures. The underlying data model is a complete realization of the CityGML building model for the levels of detail 1 to 4. Buildings can be represented by simple, monolithic objects or consist of a deep aggregation of building parts. Extensions of buildings, like balconies and stairs, can be classified thematically and provided with attributes just as single surfaces can be. LoD4 completes a LoD3 model by adding interior structures for 3D objects. For example, buildings are composed of rooms, interior doors, stairs, and furniture. This allows among other things to select the floor space of a building, so that it can later be used e.g. to derive SmartBuildings or to form 3D solids by extrusion [Döllner et al. 2005].

Buildings can be assigned addresses that are also stored in the 3D geodatabase. The implementation refers to the OASIS xAL Standard, which fits the different countries into a unified format based on XML. In order to model whole complexes of buildings, single buildings can be aggregated to form special building groups.

- **Representation of generic and prototypical 3D objects:** Generic objects enable storage and management of 3D geoobjects that are not explicitly modelled in CityGML yet, for example tunnels, or that are available in a proprietary file format only. This way, files from other software systems like architecture or graphics programs can be imported directly into the database (without interpretation). However, application systems that would like to use these data must be able to interpret the corresponding file formats after retrieving them back from the 3D geodatabase.

Prototypical objects are used for memory-efficient management of objects that occur frequently in the city model and that do not differ with respect to geometry and appearance. Examples are elements of street furniture like lanterns, road signs or benches as well as vegetation objects like shrubs, certain tree types etc. Every instance of a prototypical object is represented by a reference to the prototype, a base point and a transformation matrix for scaling, rotating and translating the prototype.

The geometries (and appearances like textures, colours etc.) of generic objects as well as prototypes can be saved either as Oracle Spatial objects or in proprietary file formats. In the latter case a single file may be saved for every object, but the file type (MIME type), the coordinate transformation matrix that is needed to integrate the object into the world CRS as well as the target CRS have to be specified.

- **Extendable object attribution:** All objects in the 3D geodatabase can be complemented with an arbitrary number of additional generic attributes. This way, it is possible to add other thematic information as well as other spatial properties to the objects at any time. In combination with the concept of generic 3D objects this provides a highly flexible storage option for object types which are not explicitly defined in the data model. Every additional attribute consists of a triple of attribute name, data type, and value. Supported data types are: String, integer and floating-point numbers, date, time, binary object (BLOB, e.g. for storing a file), Oracle Spatial geometry, texturizable 3D bodies, and 3D surfaces (aggregates).
- **Free, also recursive grouping of geoobjects:** Geoobjects can be grouped arbitrarily. The aggregates can be named and may also be provided with an arbitrary number of attributes (see above). Object groups may also contain object groups which leads to

nested aggregations of arbitrary depth. In addition, for every object of an aggregation, its role in the group can be specified explicitly (qualifiable association).

- **External references for all geoobjects:** All geoobjects can be provided with an arbitrary number of references to corresponding objects in external data sources (i.e. hyperlinks). In case of buildings this allows to store e.g. the IDs of the corresponding German digital cadastral information system ALK (Automated Real Estate Map) or future ALKIS objects (Official Cadastral Information System). Each reference consists of the external dataset's name (e.g. ALK) and the corresponding object-ID.
- **Flexible 3D geometries:** The geometry of most 3D objects can be represented through the combination of solids and surfaces as well as any - also recursive - aggregation of these elements. Surfaces may have attached different textures and colours on both their front and back face. They may also comprise information on transparency. Additional geometry types (any Oracle Spatial geometry) can be added to the geoobjects by using generic attributes.
- **Tool for importing and exporting CityGML data:** The realization of a new database administration tool allows for importing and exporting 3D city models in CityGML format according to the current specification version 1.0.0 as well as the previous version 0.4.0. Main features concern the import and export of selected Oracle objects (single objects, objectgroups, objects selected according to their spatial extent, time stamp, or feature class, etc.). The tool allows processing of very large datasets (even >> 4 GB), even if they include XLinks between CityGML features or XLinks to 3D GML geometry objects. Multiprocessor systems or multikernel CPUs are supported to speed up algorithms for decomposition and construction of complex XML-structures, leading to high performance database access.
- **Tool for importing and exporting raster data:** The tool for import and export of raster-DTMs and aerial photographs which was developed in cooperation of the company lat/lon and the University of Bonn is kept unchanged for the current version of the Berlin database. With the help of this program arbitrary regions of DTMs and aerial photographs stored in the database can be exported as seamless georeferenced TIFF pictures.
- **Implementation on the basis of Oracle 10G R2 Spatial:** For the representation of all vector and grid geometry the built-in data types provided by Oracle Spatial are used exclusively – except for multi-geometry types. This way, project-related special solutions are avoided. Besides, different geoinformation and architecture software systems enable direct access on geometry objects saved in Oracle. Therefore, these programs may directly access the data in the 3D geodatabase.
- **Open Source:** The entire software development is freely accessible to the interested public. The release was carried out under the LGPL Version 3. See the GNU Lesser General Public License at <http://www.gnu.org/licenses/lgpl-3.0.html> for more details.

1.3 System and design decisions

The 3D Geodatabase Berlin is implemented using Oracle Spatial 10G R2. In addition to the general advantages that arise from the usage of a commercial and widespread relational database management system (RDBMS), Oracle Spatial 10G R2 offers some important performance characteristics that allow an efficient implementation of the required functionalities:

- Oracle Spatial 10G R2 supports spatial data types with coordinates ranging from 2D to 4D. Even though up to now most operations and selection filters incorporate only the first two dimensions of the geometry coordinates, the supported spatial 3D indexes are sufficient for the most often required selection criteria. Furthermore, the spatial data type is supported by a number of commercial GIS that provide a database connection as for example ESRI's ArcGIS/ArcSDE or Safe Software's Feature Manipulation Engine (FME). This enables such systems to directly access the data stored in the 3D geodatabase.
- The version 10G R2 of the Oracle RDBMS offers a range of methods for efficient management of extensive georeferenced grid data. Thus it is possible to store the whole DTM as well as the whole aerial image of a city as homogeneous objects without tiling.
- The Workspace Manager provided by Oracle is a comprehensive tool for version and history management. It works widely transparent for applications connected to the database. Please note that for error-free operation all appropriate patches for the Oracle RDBMS have to be installed.
- In the future, rules can be implemented using stored procedures and trigger mechanisms which propagate updates of objects to likewise affected objects in the database (transparent for the user).

After the decision to use Oracle Spatial 10G R2 the following additional design decisions were made:

- The projection of the object-oriented data model onto an Oracle database is implemented via a relational schema. Except for the utilization of the Oracle Spatial data types, no additional object-relational modelling options for Oracle are employed, because in the version 10G R2 these cannot be used in combination with the Oracle Workspace Manager. Another reason for a purely relational model is the potential to directly connect the 3D geodatabase to commercial GIS like ArcGIS (using ArcSDE). These systems support relational database structures only. Also a relational database schema will be easier to migrate to other RDBMS like PostGIS in the future.
- Tiled, homogeneous structured grid data (aerial photographs, digital terrain models) are aggregated to form a single overall raster data object in each case. More concrete, this means that from all aerial images or DTM tiles a single raster object is generated. This approach permits the efficient and seamless output of arbitrary geographic regions by the use of built-in Oracle RDBMS functions. Thus, users of the geodatabase do not need to deal with the tiling introduced during the process of data acquisition.
- The data type GeoRaster, used for the management of grid data in Oracle, is implemented using an object-relational model. Therefore, GeoRaster objects (at present) cannot be put under the control of the version management (see above). This is why aerial photographs as well as grid-based digital terrain models are not yet versionable. Indeed, aerial images and DTMs can be changed, but these updates immediately affect all versions of the city model. However, from the point of memory efficiency versionizing of grid data is problematic anyway, because every (even a very small) amendment of an aerial image or a DTM results in a version copy of an object of several gigabytes.

2 Data modelling

2.1 UML diagram of the 3D city model

In this section the slightly simplified data model with respect to CityGML is described at the conceptual level using UML class diagrams. These form the basis for the implementation-dependent realization of the model with a relational database system which is presented in section 2.2. However, UML diagrams may also form the basis for other implementations e.g. for the definition of an exchange format based on XML or GML. The UML diagrams of the 3D city model are depicted in figures 2 to 4, 6, 9, 11, 12, 14, 16, and 18.

2.1.1 Simplifications compared to CityGML 1.0.0 / 0.4.0

CityGML is a common information model for 3D urban objects and provides a comprehensive and extensible representation of the objects. It is explained in detail in the CityGML specification [Gröger et al., 2008, Gröger et al. 2007] and [Kolbe 2009]. An analysis of the recent Berlin database indicated that for the data collected and processed a less complex schema is sufficient. Using a simplified schema usually allows to improve system performance. Therefore the first task was related to design aspects with respect to adjusting the comprehensive CityGML features to the project's needs. As result a simplified database schema was generated, allowing an optimized workflow and guaranteeing efficient processing time. The related UML-diagrams were discussed and coordinated with the project partners and translated into the relational schema. Based on this work the SQL scripts for setting up the Oracle database were generated. Please note, that all test datasets from the CityGML homepage (and others) can still be stored and managed without restrictions with this simplified data model. In fact, we have not received any CityGML dataset yet that could not be managed within this 3D geodatabase.

2.1.1.1 Multiplicities, cardinalities and recursions

Simplifications with respect to the CityGML specification were made as follows:

- **Multiplicities of attributes**

Attributes with a variable amount of occurrences (*) are substituted by a data type enabling the storage of arbitrary values (e.g. data type String with a predefined separator) or by an array with a predefined amount of elements representing the number of objects that participate in the association. This means that object attributes can be stored in a single column.

- **Cardinalities and types of relationships**

n:m relations require an additional table in the database. This table consists of the primary keys of both elements' tables which form a composite primary key. If the relation can be restricted to a 1:n or n:1 relationship the additional table can be avoided.

Therefore all n:m relations in CityGML were checked for a more restrictive definition. This results in simplified cardinalities and relations. The former n:m aggregation between rooms and furniture for example was changed to a 1:n composition, since furniture should not exist without the appropriate rooms.

- **Simplified treatment of recursions**

Some recursive relations are used in the CityGML data model. Recursive database queries may cause high cost, especially if the amount of recursive steps is unknown. In order to guarantee good performance, implementation of recursive associations receive two additional columns which contain the ID of the parent and of the root element. For example, if all building parts related to a specific building are queried, only those tuples containing the ID of the building as root element have to be selected. Thus, typical queries concerning object geometry remain high-performance.

2.1.1.2 Data type adaptation

Data types specified in CityGML were substituted by data types which allow an effective representation in the database. Strings for example are used to represent code types and number vectors; GML geometry types were changed to the Oracle-specific data type SDO_GEOMETRY. Matrices are stored each one as String (VARCHAR2), with values listed in a row-major sequence separated by spaces.

2.1.1.3 Project specific classes and class attributes

The previous version of the 3D city model of Berlin also contains classes for the representation of orthophotos as well as project specific metadata, version control and attributes for representation of addresses. Since this information is represented in the CityGML specification differently or even not at all, appropriate classes and class attributes are added or respectively adopted.

2.1.1.4 Simplified design of GML geometry classes

Spatial properties of features are represented by objects of GML3's geometry model based on the ISO 19107 standard 'Spatial Schema' [Herring 2001], representing 3D geometry according to the well-known *Boundary Representation* (B-Rep, cf. [Foley et al. 1995]). Actually only a subset of the GML3 geometry package is used. Moreover, for 2D and 3D surface-based geometry types a simpler but equally powerful model is used: These geometries are stored as polygons, which are aggregated to MultiSurfaces, CompositeSurfaces, TriangulatedSurfaces, Solids, MultiSolids, as well as CompositeSolids.

2.2 UML class diagram

The following pages cite several parts of the CityGML specification [Gröger et al., 2008] which are necessary for a better understanding. Main focus is put on explaining the customization and the differences to the CityGML standard.

Design decisions in the model are explicitly visualised within the UML diagrams. Following models are presented in detail:

- Geometric-topological model
- Appearance model
- Thematic Model
 - CityGML Core
 - Digital Terrain Model
 - Building model
 - Water bodies
 - Transportation objects
 - Vegetation objects
 - City furniture
 - Land use
 - Generic objects and attributes

For intuitive understanding, classes which will be merged to a single table in the relational schema, are shown as **orange blocks** in the UML diagrams. N to m relations, which only can be represented by additional tables, are represented as **green blocks**. Blue coloured blocks indicate non-compliant CityGML classes, which are already part of the previous version of the database and are retained for backwards compatibility purposes.

2.2.1.1 Spatial Model

2.2.1.1.1 Geometric-topological Model

The geometry model of CityGML consists of primitives, which may be combined to form complexes, composite geometries or aggregates. A zero-dimensional object is modelled as a *Point*, a one-dimensional as a *_Curve*. A curve is restricted to be a straight line, thus only the GML3 class *LineString* is used.

Combined geometries can be aggregates, complexes or composites of primitives (see illustration in figure 1). In an *Aggregate*, the spatial relationship between components is not restricted. They may be disjoint, overlapping, touching, or disconnected. GML3 provides a special aggregate for each dimension, a *MultiPoint*, a *MultiCurve*, a *MultiSurface* or a *MultiSolid*. In contrast to aggregates, a *Complex* is topologically structured: its parts must be disjoint, must not overlap and are allowed to touch, at most, at their boundaries or share parts of their boundaries. A *Composite* is a special complex provided by GML3. It can only contain elements of the same dimension. Its elements must be disjoint as well, but they must be topologically connected along their boundaries. A *Composite* can be a *CompositeSolid*, a *CompositeSurface*, or *CompositeCurve*.

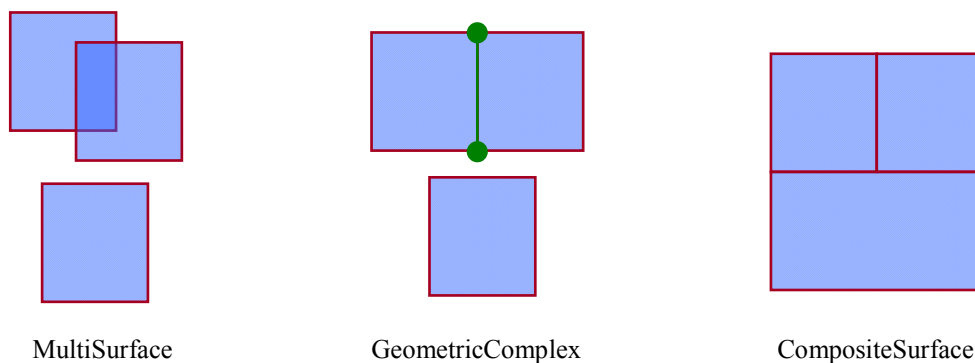


Figure 1: Different types of aggregated geometries
(from [Gröger et al., 2008])

As mentioned before, for setting up Berlin's database, the modelling of two-dimensional and three-dimensional geometry types is handled in a simplified way. All surface-based geometries are stored as polygons, which are aggregated to MultiSurfaces, CompositeSurfaces, TriangulatedSurfaces, Solids, MultiSolids, as well as CompositeSolids accordingly. This simplification substitutes the more complex representation used for those GML geometry classes in grey blocks in Figure 2. Mapping the UML diagram to the relational schema now requires only one table (SURFACE_GEOMETRY), which is explained in chapter 2.3.2.2

For the representation of textured surfaces, the Appearance model is used. You will find detailed information in chapter 2.2.1.2.

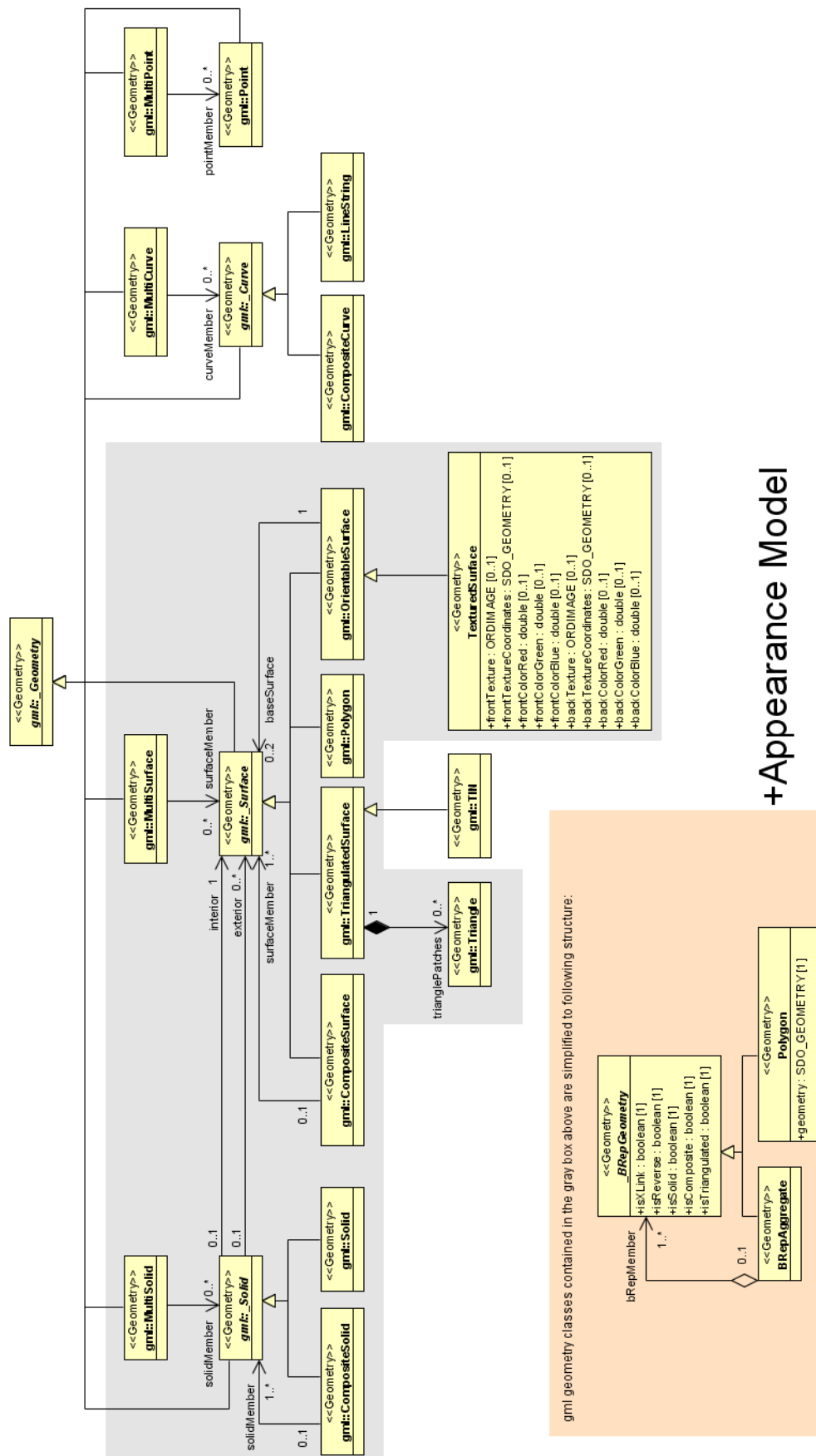


Figure 2: Geometrical-topographical model

For simplification the geometry classes in the grey block are substituted by the construct in the orange block

In order to implement topology, CityGML uses the XML concept of *XLinks* provided by GML. Each geometry object that should be shared by different geometric aggregates or different thematic features is assigned a unique identifier, which may be referenced by a GML geometry property using a *href* attribute. The XLink topology is simple and flexible and nearly as powerful as the explicit GML3 topology model. However, a disadvantage of the XLink topology is that navigation between topologically connected objects can only be performed in one direction (from an aggregate to its components), not (immediately) bidirectional, as it is the case for GML's built-in topology.

2.2.1.1.2 Implicit Geometry

The concept of implicit geometries is an enhancement of the GML3 geometry model.

An implicit geometry is a geometric object, where the shape is stored only once as a prototypical geometry, for example a tree or other vegetation objects, a traffic light or traffic sign. This prototypic geometry object is re-used or referenced many times, wherever the corresponding feature occurs in the 3D city model. Each occurrence is represented by a link to the prototypic shape geometry (in a local Cartesian coordinate system), by a transformation matrix that is multiplied with each 3D coordinate of the prototype, and by an anchor point denoting the base point of the object in the world coordinate reference system. The concept of implicit geometries is similar to the well-known concept of *primitive instancing* used for the representation of *scene graphs* in the field of computer graphics [Foley et al. 1995].

Implicit geometries may be applied to features from different thematic fields in order to geometrically represent the features within a specific level of detail (LOD). Thus, each extension module may define spatial properties providing implicit geometries for its thematic classes.

The shape of an *ImplicitGeometry* can be represented in an external file with a proprietary format, e.g. a VRML file, a DXF file, or a 3D Studio MAX file. The reference to the implicit geometry can be specified by an URI pointing to a local or remote file, or even to an appropriate web service. Alternatively, a GML3 geometry object can define the shape. This has the advantage that it can be stored or exchanged inline within the CityGML dataset. Typically, the shape of the geometry is defined in a local coordinate system where the origin lies within or near to the object's extent. If the shape is referenced by an URI, also the MIME type of the denoted object has to be specified (e.g. "model/vrml" for VRML models or "model/x3d+xml" for X3D models).

The implicit representation of 3D object geometry has some advantages compared to the explicit modelling, which represents the objects using absolute world coordinates. It is more space-efficient, and thus more extensive scenes can be stored or handled by a system. The visualisation is accelerated since 3D graphics hardware supports the scene graph concept. Furthermore, the usage of different shape versions of objects is facilitated, e.g. different seasons, since only the library objects have to be exchanged.

2.2.1.2 Appearance Model

Information about a surface's appearance, i.e. observable properties of the surface, is considered an integral part of virtual 3D city models in addition to semantics and geometry. Appearance relates to any surface-based theme, e.g. infrared radiation or noise pollution, not just visual properties and can be represented by – among others – textures and georeferenced textures. Appearances are supported for an arbitrary number of themes per city model. Each LoD of a feature can have individual appearances. Each city object or city model respectively may store its own appearance data. Therefore, the base classes *_CityObject* and *CityModel* contain a relation *appearanceMember*.

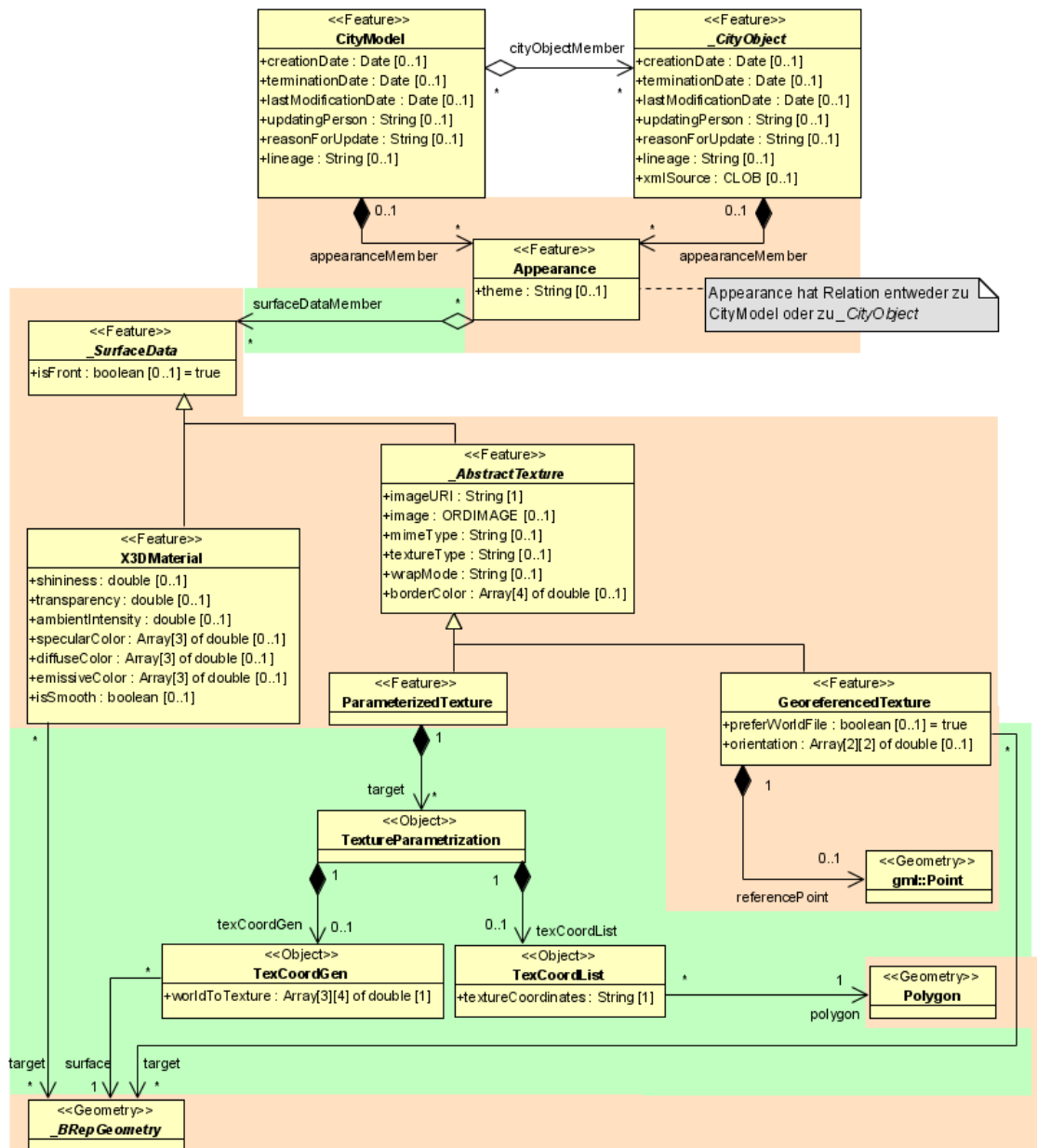


Figure 3: Appearance model

Themes are represented by an identifier only. The appearance of a city model for a given theme is defined by a set of objects of class *Appearance*, referencing this theme through the attribute *theme*. All appearance objects belonging to the same theme compose a virtual group. An *Appearance* object collects surface data relevant for a specific theme through the relation *surfaceDataMember*. Surface data is represented by objects of the abstract class *_SurfaceData*. Its only attribute is the Boolean flag *isFront*, which determines the side (front and back face of the surface) a surface data object applies to.

A constant surface property is modelled as material. A surface property, which depends on the location within the surface, is modelled as texture. Each surface object can have both a material and a texture per theme and side. This allows for providing both a constant approximation and a complex measurement of a surface's property simultaneously. If a surface object is to receive multiple textures or materials, each texture or material requires a separate theme. The mixing of themes or their usage is not explicitly defined but left to the application.

Materials define light reflection properties being constant for a whole surface object. The definition of the class *X3DMaterial* is adopted from the X3D and COLLADA specification (cf. X3D, COLLADA specification):

- *diffuseColor* defines the colour of diffusely reflected light.
- *specularColor* defines the colour of a directed reflection.
- *emissiveColor* is the colour of light generated by the surface.

All colours use RGB values with red, green, and blue channels, each defined as value between 0 and 1. Transparency is stored separately using the *transparency* element where 0 stands for fully opaque and 1 for fully transparent. *ambientIntensity* specifies the minimum percentage of *diffuseColor* that is visible regardless of light sources. *shininess* controls the sharpness of the specular highlight. 0 produces a soft glow while 1 results in a sharp highlight. *isSmooth* gives a hint for normal interpolation. If this Boolean flag is set to true, vertex normals should be used for shading (Gouraud shading). Otherwise, normals should be constant for a surface patch (flat shading). Target surfaces of the type *_BRepGeometry* are linked using the *target* relation.

The base class for textures is *_AbstractTexture*. Here, textures are always raster-based 2D textures. The raster image is specified by *imageURI* using a URI and may contain an arbitrary image data resource, even a preformatted request for a web service. The image data format can be defined using standard MIME types in the *mimeType* element. Textures can be qualified by the attribute *textureType*, differentiating between textures, which are specific for a certain object (*specific*) and prototypic textures being typical for that object surface (*typical*). Textures may also be classified as *unknown*. The specification of texture wrapping is adopted from the COLLADA standard. Possible values of the attribute *wrapMode* are *none*, *wrap*, *mirror*, *clamp* and *border*.

_AbstractTexture is further specialised according to the texture parameterisation, i.e. the mapping function from a location on the surface to a location in the texture image. Texture parameterisation uses the notion of texture space, where the texture image always occupies of the region $[0,1]^2$ regardless of the actual image size or aspect ratio. The lower left image corner is located at the origin. To receive textures, the mapping function must be known for each surface object.

The class *GeoreferencedTexture* describes a texture that uses a planimetric projection. Such a texture has a unique mapping function which is usually provided with the image file (e.g. georeferenced TIFF) or as a separate ESRI world file. The search order for an external georeference is determined by the Boolean flag *preferWorldFile*. Alternatively, inline specification of a georeference similar to a world file is possible. This internal georeference specification always takes precedence over any external georeference. *referencePoint* defines the location of the centre of the upper left image pixel in world space and corresponds to values 5 and 6 in an ESRI world file. Since *GeoreferencedTexture* uses a planimetric projection, *referencePoint* is two-dimensional and the *orientation* defines the rotation and scaling of the image in form of a 2x2 matrix (a list of 4 doubles in row-major order corresponding to values 1, 3, 2, and 4 in an ESRI world file). The CRS of this transformation is identical to the *referencePoint*'s CRS. If neither an internal nor an external georeference is given, the *GeoreferencedTexture* is invalid. Each target surface object is specified by a *target* relation. All target surface objects share the mapping function defined by the georeference.

The class *ParameterizedTexture* describes a texture with a target-dependent mapping function. The mapping is defined by associated classes of *_TextureParameterization*:

- *TexCoordList* for the concept of texture coordinates, defining an explicit mapping of a surface's boundary points to points in texture space, and
- *TexCoordGen* when using a common 3x4 transformation matrix from world space to texture space, specified by the attribute *worldToTexture*.

The related surface objects are linked by the relations *polygon* and *surface* respectively.

2.2.1.3 Thematic model

The thematic model consists of the class definitions for the most important types of objects within virtual 3D city models. Most thematic classes are (transitively) derived from the basic classes *Feature* and *FeatureCollection*, the basic notions defined in ISO 19109 and GML3 for the representation of spatial objects and their aggregations. Features contain spatial as well as non-spatial attributes, which are mapped to GML3 feature properties with corresponding data types. Geometric properties are represented as associations to the geometry classes described in chapter 2.2.1.1. The thematic model also comprises different types of interrelationships between *Feature* classes like aggregations, generalizations, and associations.

The aim of the explicit modelling is to reach a high degree of semantic interoperability between different applications. By specifying the thematic concepts and their semantics along with their mapping to UML and GML3, different applications can rely on a well-defined set of *Feature* types, attributes, and data types with a standardised meaning or interpretation. In order to allow also for the exchange of objects and/or attributes that are not explicitly modelled in CityGML, the concepts of *GenericCityObjects* and *GenericAttributes* have been introduced.

2.2.1.3.1 Core Model

The base class of all thematic classes within CityGML's data model is the abstract class *_CityObject*. *_CityObject* provides a creation and a termination date for the management of histories of features as well as generic attributes and external references to corresponding objects in other data sets. *_CityObject* is a subclass of the GML class *Feature*, thus it may inherit multiple names from *Feature*, which may be optionally qualified by a *codeSpace*. This enables the differentiation between, for example, an official name from a popular name or names in different languages (c.f. the name property of GML objects, Cox et al., 2004). The generalisation property *generalizesTo* of *_CityObject* may be used to relate features, which represent the same real-world object in different LoD, i.e. a feature and its generalized counterpart(s). The direction of this relation is from the feature to the corresponding generalised feature.

Features of *_CityObject* and its specialized subclasses may be aggregated to a *CityModel*, which is a feature collection with optional metadata. Generally, each feature has the attributes *class*, *function*, and *usage*, unless it is stated otherwise. The *class* attribute describes the classification of the objects, e.g. road, track, railway, or square. The attribute *function* contains the purpose of the object, like national highway or county road, while the attribute *usage* defines whether an object is e.g. navigable or usable for pedestrians. Here, all three attribute types are stored in String elements. Since the attributes *usage* and *function* may be used multiple times, storing them in only one String requires a single white space as unique separator. Furthermore, for each feature the geographical extent can be defined using the *Envelope* element. Minimum and maximum coordinate values have to be assigned to opposite corners of the feature's bounding box.

The subclasses of *_CityObject* comprise the different thematic fields of a city model, in the following covered by separate thematic models: building model (*_AbstractBuilding*), city furniture model (*CityFurniture*), digital terrain model (*ReliefFeature*), land use model (*LandUse*), transportation model (*TransportationComplex*), vegetation model (*_VegetationObject*), water bodies model (*WaterBody*) and generic city object model (*GenericCityObject*). The latter one allows for the modelling of features, which are not explicitly covered by one of the other models. The separation into these models strongly correlates with CityGML's extension modules, each defining a respective part of a virtual 3D city model.

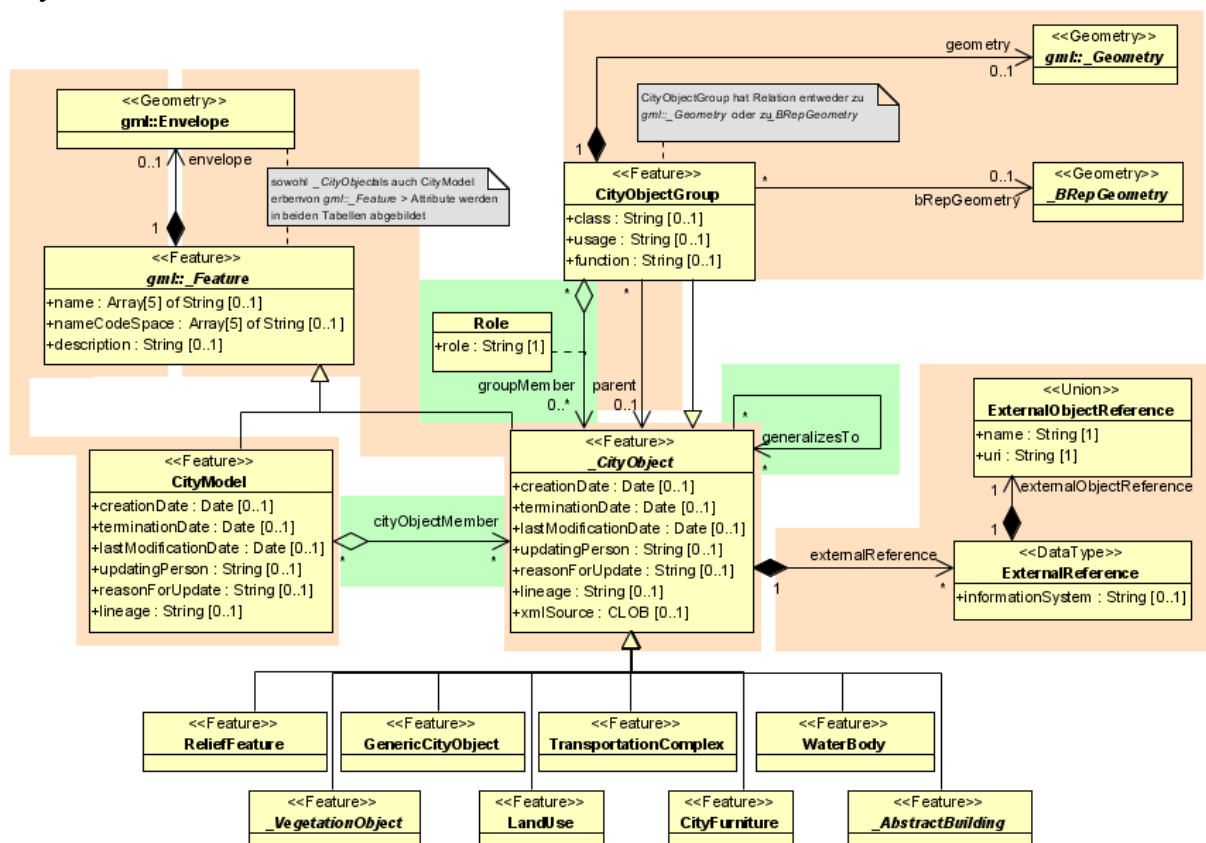


Figure 4: Core Model and thematic top level classes

3D objects are often derived from or have relations to objects in other databases or data sets. For example, a 3D building model may have been constructed from a two-dimensional footprint in a cadastre data set. The reference of a 3D object to its corresponding object in an external data set is essential, if an update must be propagated or if additional data is required (like the name and address of a building's owner in a cadastral information system). In order to supply such information, each *_CityObject* may have *External References* to corresponding objects in external data sets. Such a reference denotes the external information system and the unique identifier of the object in this system.

CityObjectGroups aggregate *CityObjects* and furthermore are defined as special *CityObjects*. This implies that a group may become a member of another group realizing a recursive aggregation schema. Since *CityObjectGroup* is a feature, it has the optional attributes *class*, *function* and *usage*. The *class* attribute allows a group classification with respect to the stated function and may occur only once. The *function* attribute is intended to express the main purpose of a group, possibly to which thematic area it belongs (e.g. site, building, transportation, architecture, unknown etc.). The attribute *usage* can be used, if the object's usage differs from its function.

Each member of a group may be qualified by a role name, reflecting the role each CityObject plays in the context of the group. Furthermore, a CityObjectGroup can optionally be assigned an arbitrary geometry object. This may be used to represent a generalised geometry generated from the member's geometries. The parent association linking a CityObjectGroup to a CityObject allows for the modelling of a generic hierarchical grouping concept. This concept is used, for example, to represent storeys in buildings. See figure 4 for the simplified UML diagram.

2.2.1.3.2 Building model

Buildings can be represented in four levels of detail (LoD1 to LoD4). The building model (cf. figure 5) allows the representation of simple buildings that consist of only one component, as well as the representation of complex relations between parts of a building, e.g. a building consisting of three parts – a main house, a garage and an extension. The parts can again consist of parts etc. The subclasses *Building* and *BuildingPart* of *AbstractBuilding* enable these modelling options. In the case of a simple, one-piece house there is only one *Building* which inherits all attributes and relations from *AbstractBuilding*. However, such a *Building* can also comprise *BuildingParts* which likewise inherit all properties from *AbstractBuilding*: the building's class, function (e.g. residential, public, or industry), usage, year of construction, year of demolition, roof type, measured height, and the number and individual heights of all its storeys above and below ground (cf. figure 6). Furthermore, *Addresses* can be assigned to *Buildings* or *BuildingParts*. In particular, *BuildingParts* may again comprise *BuildingParts* as components, because the composition relation is inherited. This way a tree-like hierarchy can be created whose root object is a *Building* and whose non-root nodes are *BuildingParts*. The attribute values are generally filled in the lower hierarchy level, because basically every part can have its own construction year and function. However, the function can also be defined in the root of the hierarchy and therefore span the whole building. The individual *BuildingParts* within a *Building* must not penetrate each other and must form a coherent object.

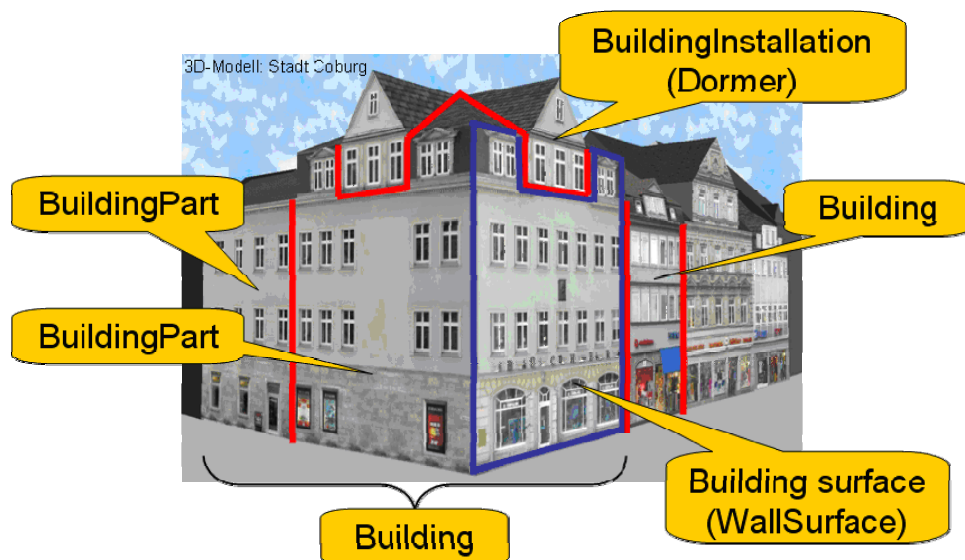


Figure 5: Example of buildings consisting of building parts
(from: [Plümer et al., 2005])

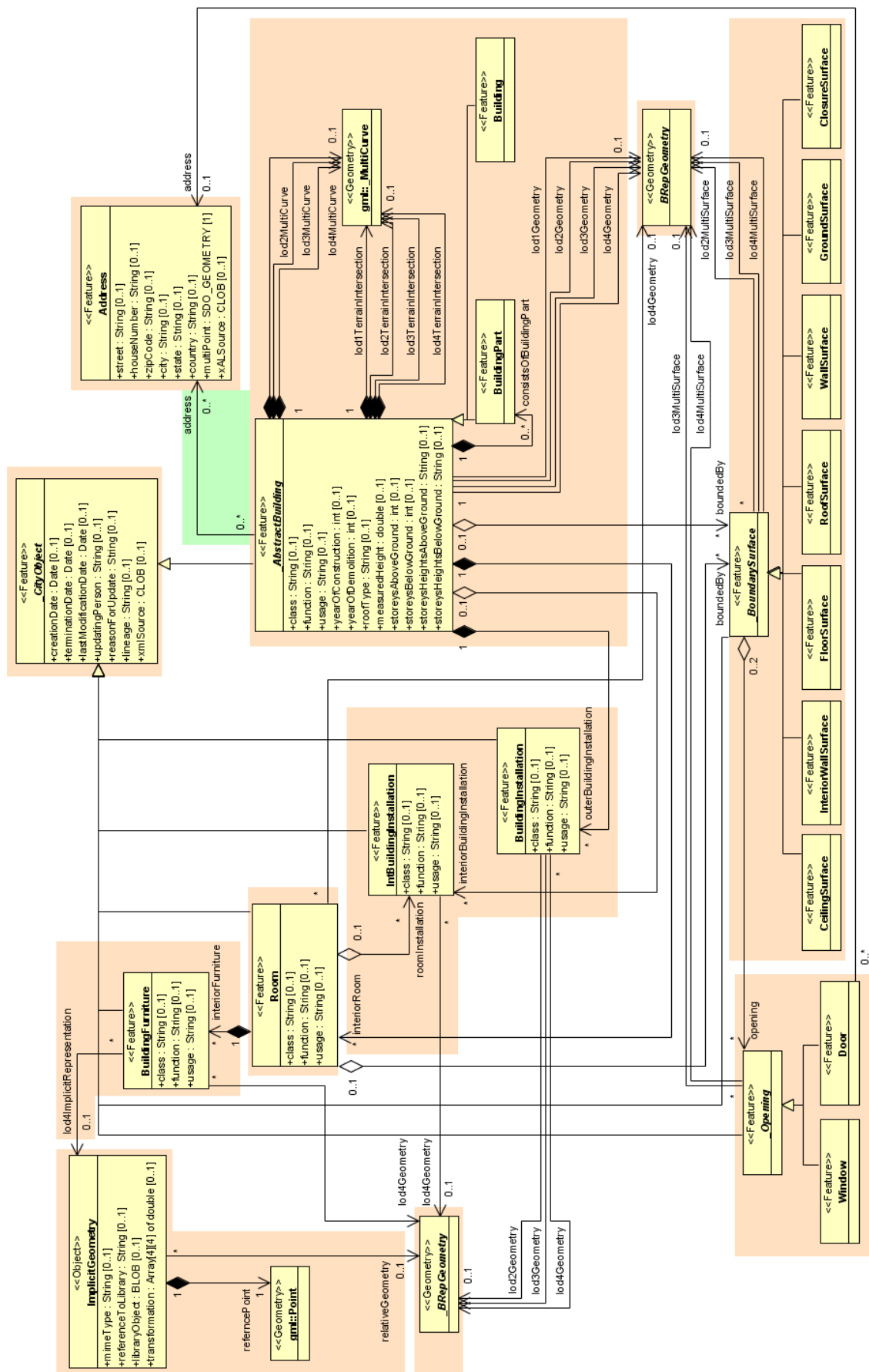


Figure 6: UML diagram of building model

The geometry of an *AbstractBuilding* is realized via a relation to the class *GM_Composite*. An individual geometry representation is provided for each of the four levels of detail, LoD1 to LoD4. This is realized via the relations *lod1Geometry*, ..., *lod4Geometry*. Therefore, a single building can have multiple spatial representations in different levels of detail at the same time.

In LoD1, a building model consists of a geometric representation of the building volume. Optionally, a *MultiCurve* representing the *TerrainIntersectionCurve* can be specified. This geometric representation is refined in LoD2 by additional *MultiSurface* and *MultiCurve* geometries, used for modelling architectural details like a roof overhang, columns, or antennas. In LoD2 and higher LoDs the outer facade of a building can also be differentiated semantically by the classes *_BoundarySurface* and *BuildingInstallation*. A *_BoundarySurface* is a part of the building's exterior shell with a special function like wall (*WallSurface*), roof (*RoofSurface*), ground plate (*GroundSurface*), or closing surface (*ClosureSurface*) as shown in figure 7. Closure surfaces can be used to virtually seal open buildings as for example hangars, allowing e.g. volume calculation. The *BuildingInstallation* class is used for building elements like balconies, chimneys, dormers, or outer stairs, strongly affecting the outer appearance of a building. A *BuildingInstallation* is used for the representation of chimneys, stairs, balconies etc. and optionally has the attributes *class*, *function*, and *usage*.

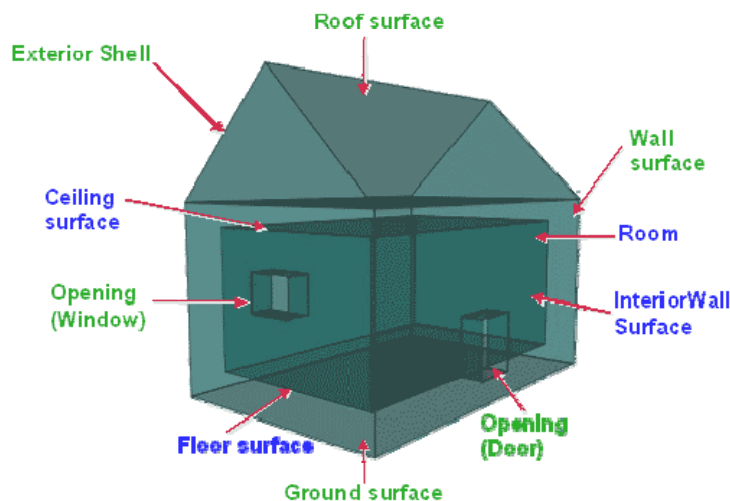


Figure 7: Boundary surfaces
(from: [Gröger et al., 2008])

In LoD3, the openings in *_BoundarySurface* objects (doors and windows) can be represented as thematic objects. In LoD4, the highest level of resolution, also the interior of a building, composed of several rooms, is represented in the building model by the class *Room*. The aggregation of rooms according to arbitrary, user-defined criteria (e.g. for defining the rooms corresponding to a certain storey) is achieved by employing the general grouping concept provided by CityGML. Interior installations of a building, i.e. objects within a building which (in contrast to furniture) cannot be moved, are represented by the class *IntBuildingInstallation*. If an installation is attached to a specific room (e.g. radiators or lamps), they are associated with the *Room* class, otherwise (e.g. in case of rafters or pipes) with *_AbstractBuilding*. A *Room* may have the attributes *class*, *function*, and *usage* referenced to external code lists. The *class* attribute allows a classification of rooms with respect to the stated function, e.g. commercial or private rooms, and occurs only once. The *function* attribute is intended to express the main purpose of the room, e.g. living room, kitchen. The attribute *usage* can be used if the object's usage differs from its function. Since the attributes *usage* and *function* may be used multiple times, storing them in only one String requires a single white space as unique separator.

The visible surface of a room is represented geometrically as a *Solid* or *MultiSurface*. Semantically, the surface can be structured into specialised *_BoundarySurfaces*, representing floor (*FloorSurface*), ceiling (*CeilingSurface*), and interior walls (*InteriorWallSurface*) (cf. figure 7). Room furniture, like tables and chairs, can be represented in the CityGML building model with the class *BuildingFurniture*. A *BuildingFurniture* may have the attributes *class*, *function*, and *usage*.

2.2.1.3.3 CityFurniture Model

City furniture objects are immovable objects like lanterns, traffic lights, traffic signs, flower buckets, advertising columns, benches, delimitation stakes, or bus stops. The class *CityFurniture* may have the attributes *class* and *function* (cf. UML-diagram, figure 8). Their possible values are explained in detail in the CityGML specification. The class attribute allows an object classification like traffic light, traffic sign, delimitation stake, or garbage can, and can occur only once. The function attribute describes to which thematic area the city furniture object belongs (e.g. transportation, traffic regulation, architecture etc.), and can occur multiple times. The hierarchy between class and function is not reflected in the external code lists. Inconsistencies have to be checked by the application tools.

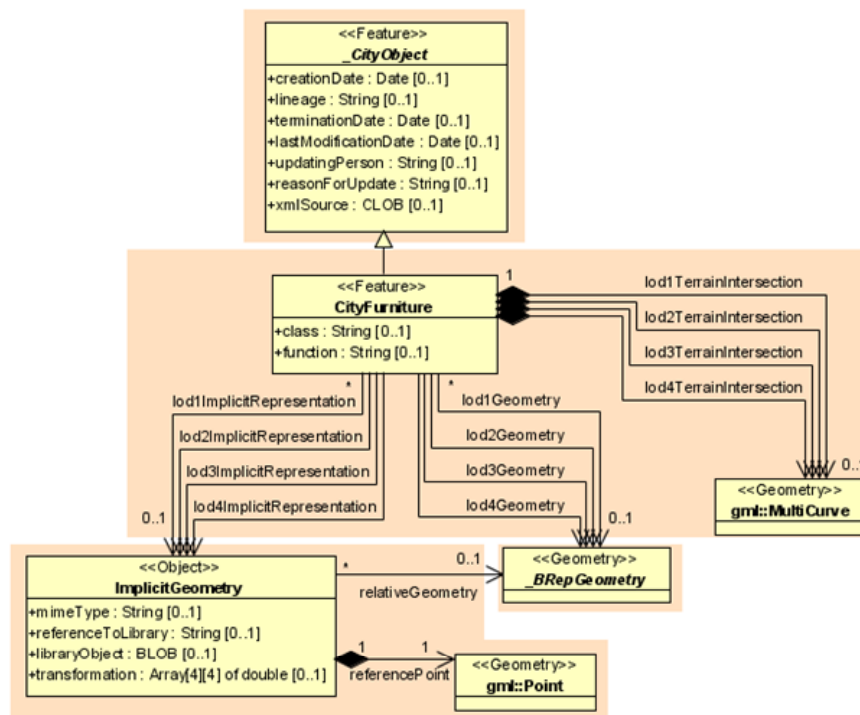


Figure 8: City furniture model

Since *CityFurniture* is a subclass of *CityObject* and hence is a feature, it inherits the attribute *gml:name*. As with any *CityObject*, *CityFurniture* objects may be assigned *ExternalReferences* and *GenericAttributes*. For *ExternalReferences* city furniture objects can have links to external thematic databases. Thereby, semantical information of the objects, which can not be modelled in CityGML, can be transmitted and used in the 3D city model for further processing, for example information from systems of power lines or pipelines, traffic sign cadastre, or water resources for disaster management.

City furniture objects can be represented in city models with their specific geometry, but in most cases the same kind of object has an identical geometry. The geometry of *CityFurniture* objects in LoD 1-4 may be represented by an explicit geometry (*lodXGeometry* where *X* is between 1 and 4) or an *ImplicitGeometry* object (*lodXImplicitRepresentation* with *X* between 1 and 4). In the concept of *ImplicitGeometry* the geometry of a prototype city furniture object

is stored only once in a local coordinate system and referenced by a number of features. Spatial information of city furniture objects can be taken from city maps or from public and private external information systems. In order to specify the exact intersection of the DTM with the 3D geometry of a city furniture object, the latter can have a *TerrainIntersectionCurve* (TIC) for each LoD. This allows for ensuring a smooth transition between the DTM and the city furniture object.

2.2.1.3.4 Digital terrain model

The city model includes a very adaptable digital terrain model (DTM) which permits the combination of heterogeneous DTM types (grid, TIN, break lines, mass points) available in different levels of detail. Compared to the previous 3D database of Berlin the DTM remains untouched.

A DTM fitting to a certain city model is represented by the class *ReliefFeature*. This is a *CityObject* having the LoD step that fits the DTM as attribute. A relief consists of several *ReliefComponents*. Each of these components that are likewise *CityObjects* also comprises a LoD step. Individual geometrical types of the components are defined by the four subclasses of *ReliefComponent*: breaklines, triangular networks (TINs), mass points, and grids (Raster). Geometrically, the corresponding ISO 19107 or GML classes define these types: breaklines by a single *MultiCurve*, TINs by *TriangulatedSurfaces*, mass points by *MultiPoint*, and Raster by *RectifiedGridCoverage*.

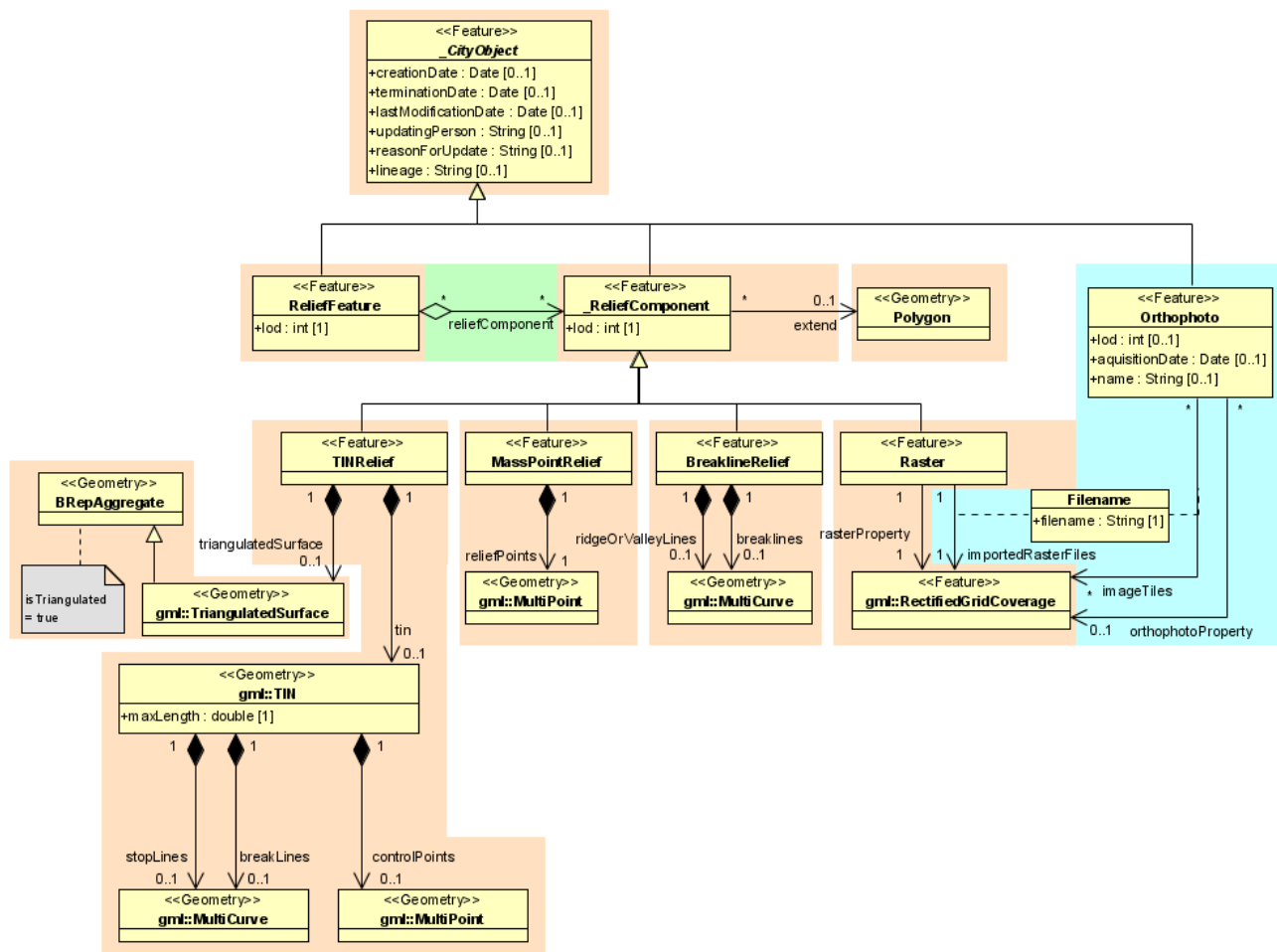


Figure 9: UML diagram representing the digital terrain model

A relief can contain ReliefComponents of heterogeneous type and different LoDs. A relief in LoD2, for example, can contain some LoD3-TIN-ReliefComponents beside a LoD2-Raster-ReliefComponent. In some cases even a LoD1 grid may exist in some regions of the relief.

A grid is defined by the GML3 class *RectifiedGridCoverage* that also allows the representation of non-axis parallel grids. However, managing grids requires attention to a peculiarity that arises from the two-stage import process. In the first step single, small grid tiles are imported. In a second step these tiles are then combined to form the overall grid. In order to interim storage of these smaller grids (tiles), the relation *importedRasterTiles* between grid and *RectifiedGridCoverage* can be used. Finally the overall grid, generated during the second import step, is constructed as *RectifiedGridCoverage*. It is linked via the relation *rasterProperty*.

In order to geometrically separate the individual components of a grid, which can exist in different LoD, the validity polygon of a component (*extent*) is used. This polygon defines the scope in which the component is valid. A grid with three components is shown in Figure 10. It depicts a coarse raster containing two high-resolution TINs (TIN 1 and 2). The validity polygon of the raster is represented by the blue line, while the validity polygons of the TINs are bordered in green and red. In this case, the validity polygon of the grid has two holes where the grid is not valid, although it does exist. Instead, the high-resolution TINs are used for the representation of the terrain in these regions. That means the validity polygons of the TINs exactly fit the two holes in the validity polygon of the grid.

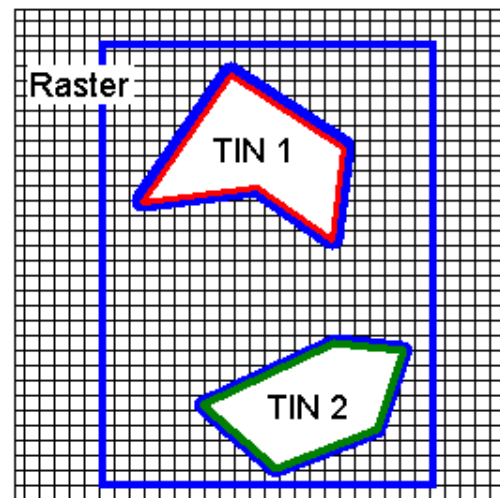


Figure 10: A relief, consisting of three components and its validity polygons (from: [Plümer et al., 2005])

In the simplest and most frequent case, the validity polygon of a grid corresponds exactly with its Bounding box, i.e. the spatial extent of the grid.

2.2.1.3.5 Generic CityObject Model

The concept of generic objects and attributes is introduced to ensure the storage and exchange of 3D objects, which are not covered by explicitly modelled classes within CityGML or which requires additional attributes. These generic extensions are realised by the classes *GenericCityObject* and *GenericAttribute* (cf. Figure 11).

A *GenericCityObject* may have the attributes *class*, *function*, and *usage* defined as *String*. The *class* attribute allows an object classification within the thematic area such as bridge, tunnel, pipe, power line, dam, or unknown. The *function* attribute describes to which thematic area the *GenericCityObject* belongs (e.g. site, transportation, architecture, energy supply, water supply, unknown etc.). The attribute *usage* can be used, if the object's usage differs from its function. Since the attributes *usage* and *function* may be used multiple times, storing them in only one string requires a single white space as unique separator. Each *_CityObject* and all thematic subclasses can have an arbitrary number of *GenericAttributes*. Data types may be *String*, *Integer*, *Double* (floating point number), *URI* (Unified Resource Identifier), *Date*, *BLOB* (for binary large objects), *_BRepGeometry* (for 2D and 3D geometries), or even

arbitrary geometries. The attribute type is defined by the selection of the particular subclass (*StringAttribute*, *IntAttribute* etc.).

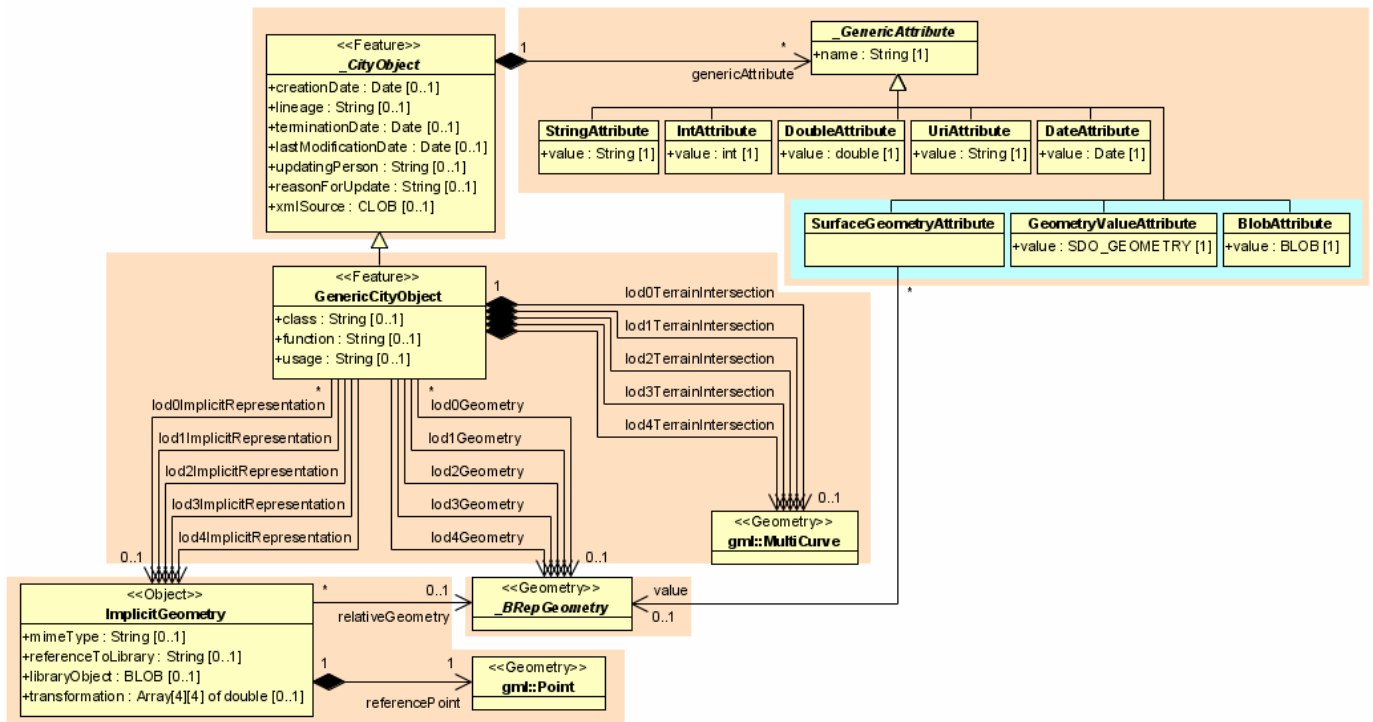


Figure 11: Generic CityObject model

The geometry of a *GenericCityObject* can either be an explicit GML3 geometry or an *ImplicitGeometry*. In the case of an explicit geometry, the object can have only one geometry for each LoD, which may be an arbitrary 3D GML geometry object (class *_Geometry*, which is the base class of all GML geometries, *lodXGeometry*, *X* in $0 \dots 4$). Absolute coordinates according to the reference system of the city model must be given for the explicit geometry. In the case of an *ImplicitGeometry*, a reference point (anchor point) of the object and optionally a transformation matrix must be given. In order to compute the actual location of the object, the transformation of the local coordinates into the reference system of the city model must be processed and the anchor point coordinates must be added. The shape of an *ImplicitGeometry* can be given as an external resource with a proprietary format, e.g. a VRML or DXF file from a local file system or an external web service. Alternatively the shape can be specified as a 3D GML3 geometry with local Cartesian coordinates using the property *relativeGeometry*.

In order to specify the exact intersection of the DTM with the 3D geometry of a *GenericCityObject*, the latter can have *TerrainIntersectionCurves* for every LoD. This is important for 3D visualization but also for certain applications like driving simulators. For example, if a bridge should be represented as a *GenericCityObject*, a smooth transition between the DTM and the road on the bridge would have to be ensured (in order to avoid unrealistic bumps).

2.2.1.3.6 LandUse Model

LandUse objects describe areas of the earth's surface dedicated to a specific land use. They can be employed to represent parcels in 3D. Figure 12 shows the UML diagram of land use objects.

Every *LandUse* object may have the attributes *class* (e.g. settlement area, industrial area, farmland etc.), *function* (purpose, e.g. cornfield), and *usage* which can be used, if the way the

object is actually used differs from the function. Since the attributes *usage* and *function* may be used multiple times, storing them in only one string requires a single white space as unique separator.

The *LandUse* object is defined for all LoD 0-4 and may have different geometries for each LoD. The surface geometry of a *LandUse* object is required to have 3D coordinate values. It must be a GML3 *MultiSurface*, which might be assigned appearance properties like material (*X3DMaterial*) and texture (*_AbstractTexture* and its subclasses).

LandUse objects can be employed to establish a coherent spatio-semantic tessellation of the earth's surface. In this case, topological relations between neighbouring *LandUse* objects should be made explicit by defining the boundary LineStrings only once and referencing them in the corresponding polygons using XLinks.

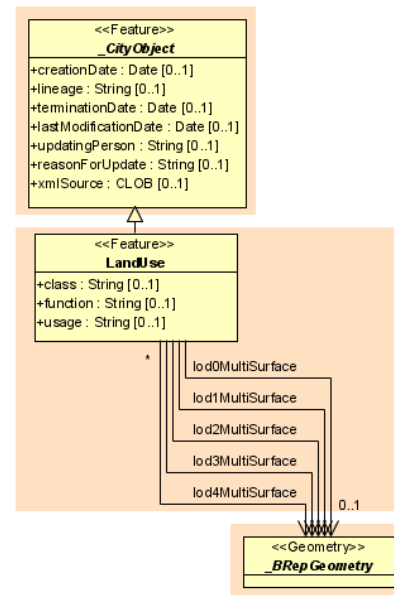


Figure 12: Landuse model

2.2.1.3.7 Transportation Model

The transportation model of CityGML is a multi-functional, multi-scale model focusing on thematic and functional as well as geometrical/topological aspects. Transportation features are represented as a linear network in LoD0. Starting from LoD1, all transportation features are geometrically described by 3D surfaces.

The main class is *TransportationComplex* (cf. Figure 14), which represents, for example, a road, a track, a railway, or a square. It is composed of the parts *TrafficArea* and *AuxiliaryTrafficArea*. Figure 13 depicts an example for a LoD2 *TransportationComplex* configuration within a virtual 3D city model. The *Road* consists of several *TrafficAreas* for the sidewalks, road lanes, parking lots, and of *AuxiliaryTrafficAreas* below the raised flower beds.

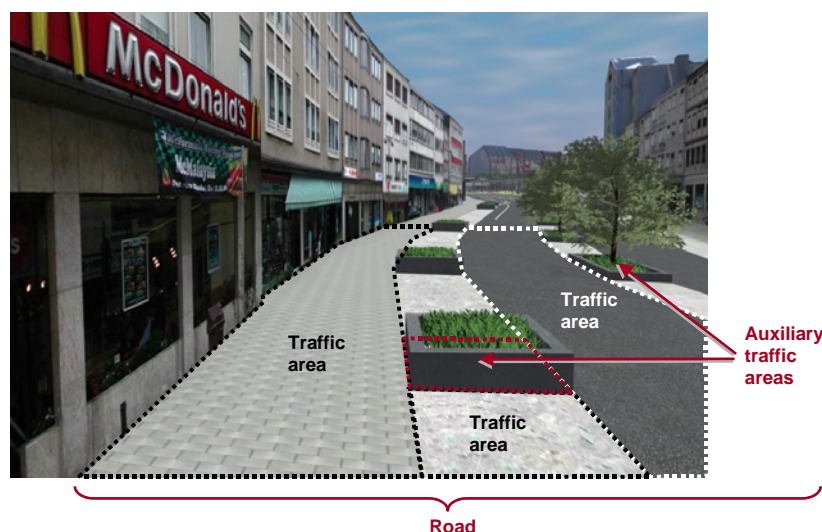


Figure 13: LoD2 representation of a transportation complex (from: [Gröger et al., 2008])

The road itself is represented as a *TransportationComplex*, which is further subdivided into *TrafficAreas* and *AuxiliaryTrafficAreas*. The *TrafficAreas* are those elements, which are important in terms of traffic usage, like car driving lanes, pedestrian zones and cycle lanes. The *AuxiliaryTrafficAreas* are describing further elements of the road, like kerbstones, middle lanes, and green areas.

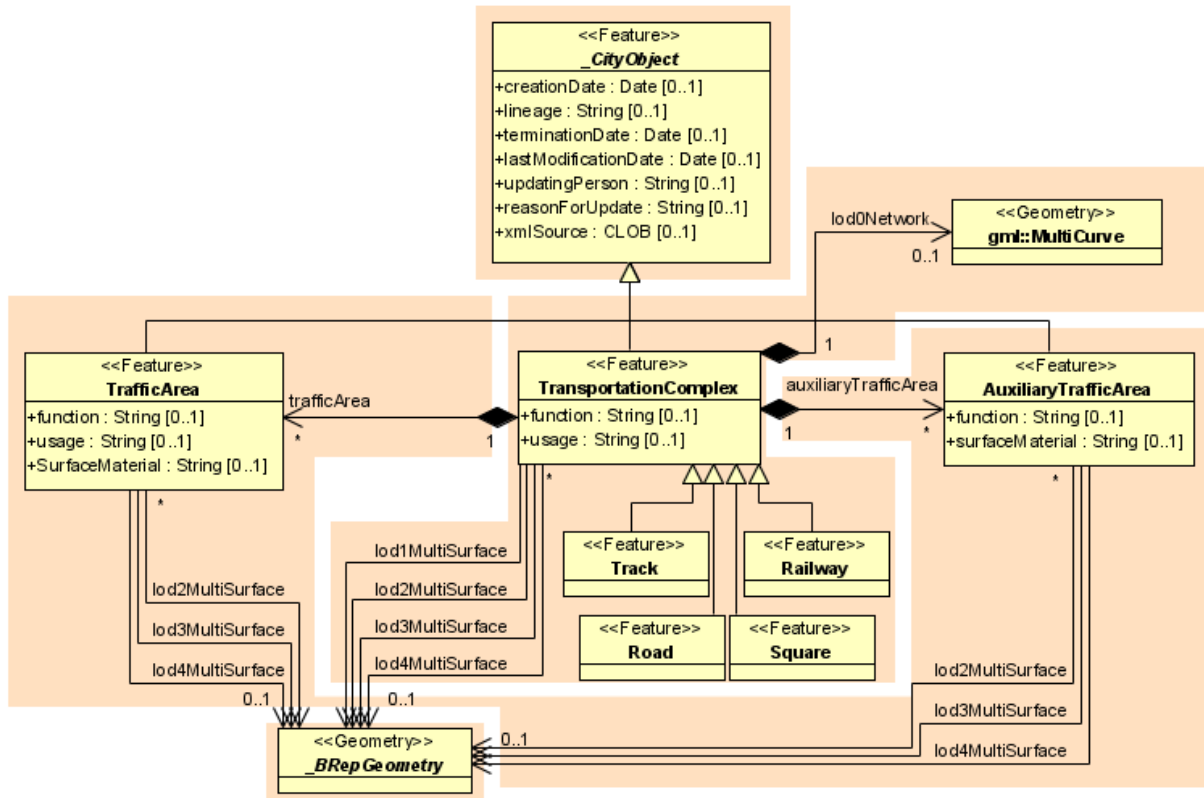


Figure 14: UML model for transportation complex

TransportationComplex objects can be thematically differentiated using the subclasses *Track*, *Road*, *Railway*, and *Square*. Every *TransportationComplex* has the attributes *function* and *usage*, referencing to the external code lists. The attribute *function* describes the purpose of the object like, for example national motorway, country road, or airport.

In addition, every *TrafficArea* may have the attributes *function*, *usage*, and *surfaceMaterial*. The *function* describes whether the object is a car driving lane, a pedestrian zone, or a cycle lane, while the *usage* attribute indicates which modes of transportation can use it (e.g. pedestrian, car, tram, roller skates). The attribute *surfaceMaterial* specifies the type of pavement and may also be used for *AuxiliaryTrafficAreas* (e.g. asphalt, concrete, gravel, soil, rail, grass etc.). The *function* attribute of the *AuxiliaryTrafficArea* defines, among others, kerbstones, middle lanes, or green areas. The possible values are specified in external code lists.

TransportationComplex is a subclass of *_CityObject*. The geometrical representation of the *TransportationComplex* varies through the different levels of detail. In the coarsest LoD0, the transportation complexes are modelled by line objects establishing a linear network. Starting from LoD1, a *TransportationComplex* provides an explicit surface geometry, reflecting the actual shape of the object, not just its centreline. In LoD2 to LoD4, it is further subdivided thematically into *TrafficAreas*, which are used by transportation, such as cars, trains, public transport, airplanes, bicycles, or pedestrians and in *AuxiliaryTrafficAreas*, which are of minor importance for transportation purposes, for example road markings, green spaces or flower tubs.

2.2.1.3.8 Vegetation Model

The vegetation model of CityGML distinguishes between solitary vegetation objects like trees and vegetation areas, which represent biotopes like forests or other plant communities. Single vegetation objects are modelled by the class *SolitaryVegetationObject*, while for areas filled with specific vegetation the class *PlantCover* is used.

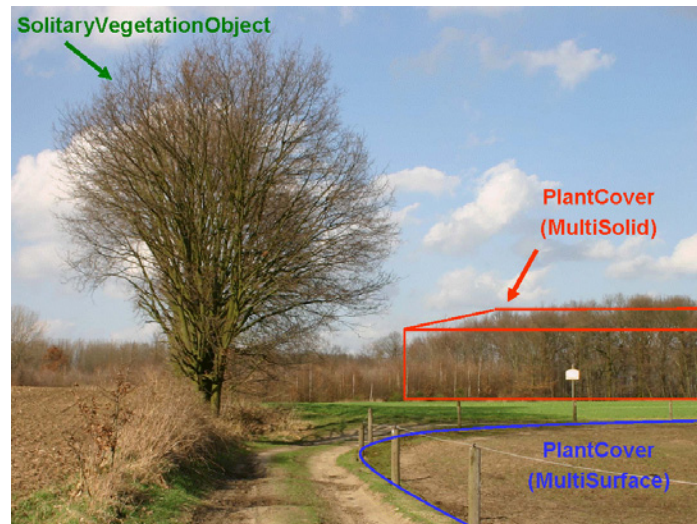


Figure 15: Image illustrates objects of the vegetation model
(from: [Gröger et al., 2008])

The geometry representation of a *PlantCover* feature may be a MultiSurface or a MultiSolid, depending on the vertical extent of the vegetation. For example regarding forests, a MultiSolid representation might be more appropriate (cf. **Figure 15**).

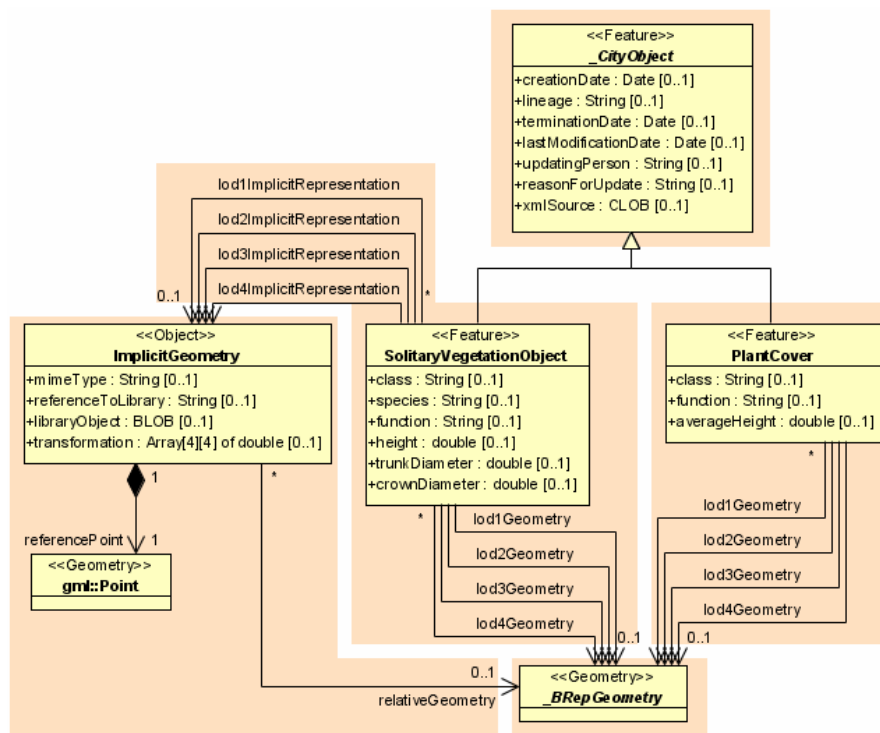


Figure 16: Vegetation Model

The UML diagram of the vegetation model is depicted in Figure 16. A *SolitaryVegetationObject* may have the attributes *class* (e.g. tree, bush, grass), *species* (species' name, e.g. *Abies alba*), and *function* (e.g. botanical monument), *height*, *trunkDiameter* and *crownDiameter*. A *PlantCover* feature may have the attributes *class* (plant community), *function* (e.g. national forest) and *averageHeight*. Since both *SolitaryVegetationObject* and *PlantCover* are *CityObjects*, they inherit all attributes of a city object, in particular its name (*gml:name*) and an *ExternalReference* to a corresponding object in an external information system, which may contain botanical information from public environmental agencies.

The geometry of a *SolitaryVegetationObject* may be defined in LoD 1-4 by absolute coordinates, or prototypically by an *ImplicitGeometry*. Season dependent appearances may be mapped using *ImplicitGeometries*. For visualisation purposes, only the content of the library object defining the object's shape and appearance has to be swapped.

A *SolitaryVegetationObject* or a *PlantCover* may have a different geometry in each LoD. Whereas a *SolitaryVegetationObject* is associated with the *_Geometry* class representing an arbitrary GML geometry (by the relation *lodXGeometry*), a *PlantCover* is restricted to be either a *MultiSolid* or a *MultiSurface*.

2.2.1.3.9 WaterBodies Model

The water bodies model represents the thematic aspects and 3D geometry of rivers, canals, lakes, and basins. In LoD 2-4 water bodies are bounded by distinct thematic surfaces. These surfaces are the obligatory *WaterSurface*, defined as the boundary between water and air, the optional *WaterGroundSurface*, defined as the boundary between water and underground (e.g. DTM or floor of a 3D basin object), and zero or more *WaterClosureSurfaces*, defined as virtual boundaries between different water bodies or between water and the end of a modelled region (cf. Figure 17). A dynamic element may be the *WaterSurface* to represent temporarily changing situations of tidal flats.

Each *WaterBody* object may have the attributes *class* (e.g. lake, river, or fountain), *function* (e.g. national waterway or public swimming) and *usage* (e.g. navigable) referencing to external code lists. Since the attributes *usage* and *function* may be used multiple times, storing them in only one string requires a single white space as unique separator.

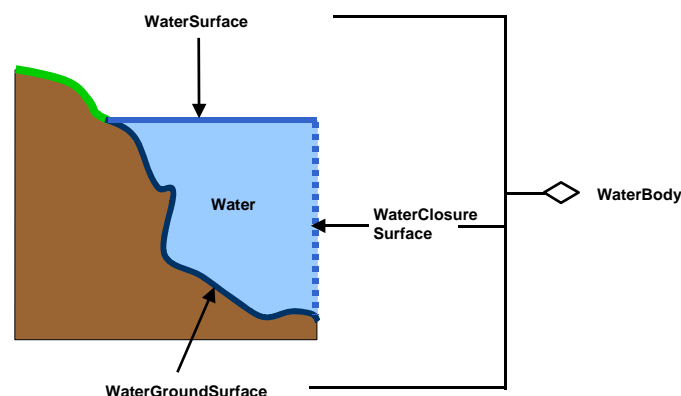


Figure 17: Definition of waterbody attributes
(from: [Gröger et al., 2008])

WaterBody is a subclass of the root class *_CityObject*. The geometrical representation of the *WaterBody* varies for different levels of detail. The *WaterBody* can be differentiated semantically by the class *_WaterBoundarySurface*. A *_WaterBoundarySurface* is a part of the water body's exterior shell with a special function like *WaterSurface*, *WaterGroundSurface* or *WaterClosureSurface*. As with any *_CityObject*, *WaterBody* objects as well as *WaterSurface*,

WaterGroundSurface, and *WaterClosureSurface* objects may be assigned *ExternalReferences* and *GenericAttributes*.

Both LoD0 and LoD1 represent a low level of illustration and high grade of generalisation. Here the rivers are modelled as *MultiCurve* geometry and brooks are omitted. Seas, oceans, and lakes with significant extent are represented as *MultiSurfaces* (cf. Figure 18).

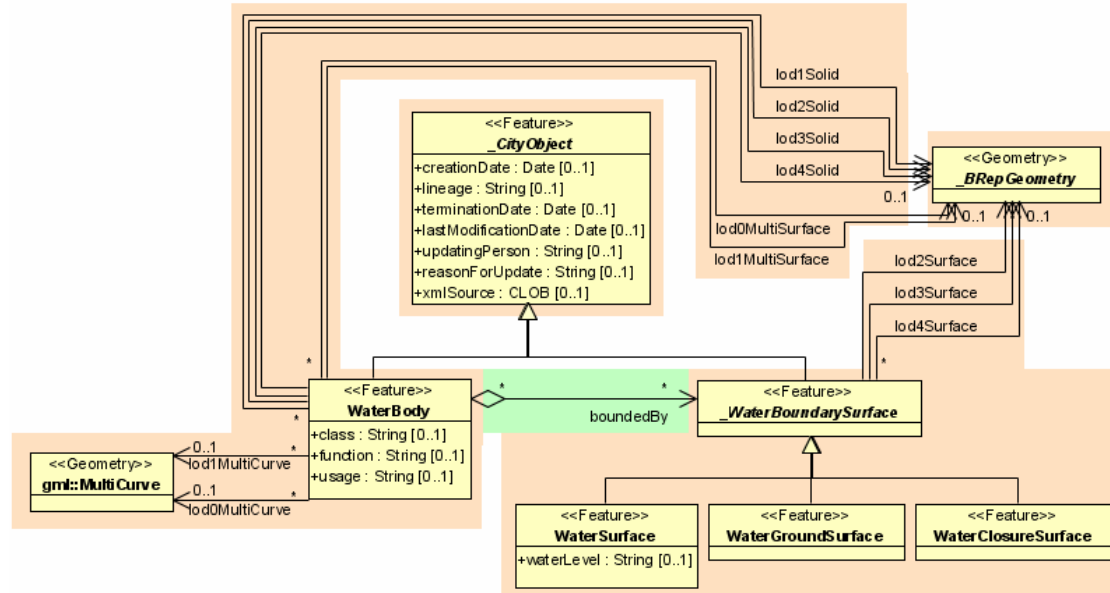


Figure 18: Waterbody model

Starting from LoD1, water bodies may also be modelled as volumes filled with water, represented by *Solids*. If a water body is represented by a *Solid* in LoD2 or higher, the surface geometries of the corresponding thematic *WaterClosureSurface*, *WaterGroundSurface*, and *WaterSurface* objects must coincide with the exterior shell of the *Solid*. This can be ensured, if for one LoD *X* the respective *lodXSurface* elements (where *X* is between 2 and 4) of *WaterClosureSurface*, *WaterGroundSurface*, and *WaterSurface* reference the corresponding polygons (using *XLink*) within the *CompositeSurface* that defines the exterior shell of the *Solid*. Furthermore, every *_WaterBoundarySurface* must have at least one associated surface geometry attached.

The water body model implicitly includes the concept of *TerrainIntersectionCurves* (TIC), e.g. to specify the exact intersection of the DTM with the 3D geometry of a *WaterBody* or to adjust a *WaterBody* or *WaterSurface* to the surrounding DTM. The rings defining the *WaterSurface* polygons implicitly delineate the intersection of the water body with the terrain or basin.

2.3 Relational database schema

2.3.1 Mapping rules, schema conventions

2.3.1.1 Mapping of classes onto tables

Generally, one or more classes of the UML diagram are mapped onto one table; the name of the table is identical to the class name (a leading underscore indicating an abstract classes is left out). Classes are combined into a single table according to the class relations as shown in the UML diagrams by using **orange coloured boxes**. The scalar attributes of the classes become columns of the corresponding table with identical name. The types of the attributes are customized to Oracle data types.

2.3.1.2 Explicit declaration of class affiliation

In the (meta) table OBJECTCLASS, all class names (attribute CLASSNAME) of the schema are managed. The relation of the subclass to its parent class is represented via the attribute SUPERCLASS_ID in the subclass as a foreign key to the ID of the parent class.

The table OBJECTCLASS is used to efficiently determine the affiliation to a class in the superclass tables. In addition, the table CITYOBJECT contains the attribute CLASS_ID which refers to the respective table OBJECTCLASS. This way, while looking at a tuple in CITYOBJECT, the subclass and – if needed – the name of the class can be determined directly.

2.3.2 Database schema

In the following paragraph, the tables of the relational schema are described in detail. In figures 19, 22, 28, 30-37 the schema is displayed graphically. The description is based on the remarks on UML charts in chapter 2.2. Focus is put on situations where the conversion into tables leads to changes in the model.

2.3.2.1 Core Model

CITYOBJECT, CITYOBJECT_SEQ

All CityObjects (and instances of the subclasses like buildings etc.) are represented by tuples in the table CITYOBJECT. The fields are identical to the attributes of the corresponding UML class. The BoundingBox (Envelope) is realized by an Oracle data type SDO_GEOMETRY that contains an axis-parallel 2D polygon (SDO_GTYPE = 2003, SDO_ETYPE = 1003 and SDO_INTERPRETATION = 3)¹.

For identification of each object a unique identifier is essential. The use of GMLIDs (attribute gml:id) only cannot ensure this, because in different CityGML files objects may use the same ID. Thus the attribute GMLID_CODESPACE is added which contains the full path to the object – typically the path of the imported CityGML file.

For performance reasons GML_NAME is not added to the table CITYOBJECT, but included in the subclasses. This helps avoiding JOIN operations during database querying, in case a specific object should be selected. If, for example, the *Brandenburger Tor* in Berlin is of interest, the GML_NAME 'Brandenburger Tor' must be used in the SELECT clause for obtaining all parts related to this site. The attributes NAME or NAME_CODESPACE can contain more than one gml:name property. In this case they have to be separated by the string

¹ In the future, we will consider to use three-dimensional BoundingVolumes.

‘--/\--’, which is used as a separator string. The CityGML exporter will then create multiple occurrences of <gml:name> elements.

The attribute CLASS_ID provides information on the class affiliation of the CityObject. This helps to identify the proper subclass tables.

The next free ID value for the table CITYOBJECT is provided by the database sequence CITYOBJ_SEQ.

CITYMODEL, CITYMODEL_SEQ

CityObject features may be aggregated to a single CityModel. A CityModel serves as root element of a CityGML feature collection. In order to provide a unique identifier in table CITYMODEL, the next available ID value is provided by the sequence CITYMODEL_SEQ.

EXTERNAL_REFERENCE, EXTERNAL_REF_SEQ

The table EXTERNAL_REFERENCE is used to store external references; the foreign key CITYOBJECT_ID refers to the associated CityObject. The sequence EXTERNAL_REF_SEQ provides the next available ID value for EXTERNAL_REFERENCE.

CITYOBJECTGROUP, GROUP_TO_CITYOBJECT

The aggregation concept described in paragraph 2.1.2.3.1 is realized by two tables. The m:n relationship between an object group (table CITYOBJECTGROUP) consisting of city objects contained in CITYOBJECT is realized by the table GROUP_TO_CITYOBJECT, which associates the IDs of both tables. Table 1 shows an example, in which two buildings are grouped to a hotel complex.

CITYOBJECTGROUP (excerpt)						
ID	NAME	NAME_CODESPACE	DESCRIPTION	CLASS	FUNCTION	USAGE
1	Hotel complex	NULL	NULL	NULL	Building group	NULL

GROUP_TO_CITYOBJECT		
CITYOBJECT_ID	CITYOBJECTGROUP_ID	ROLE
2	1	Main building
4	1	Annex

CITYOBJECT (excerpt)						
ID	CLASS_ID	GML_ID	GML_ID_CODESPACE	ENVELOPE	CREATION_DATE	TERMINATION_DATE
2	26	Build1632	NULL	SDO_GEOMETRY(...)	17.12.08	NULL
4	26	Build1633	NULL	SDO_GEOMETRY(...)	17.12.08	NULL
1	23	Group1700	NULL	NULL	17.12.08	NULL

Table 1: Cityobjectgroup tables

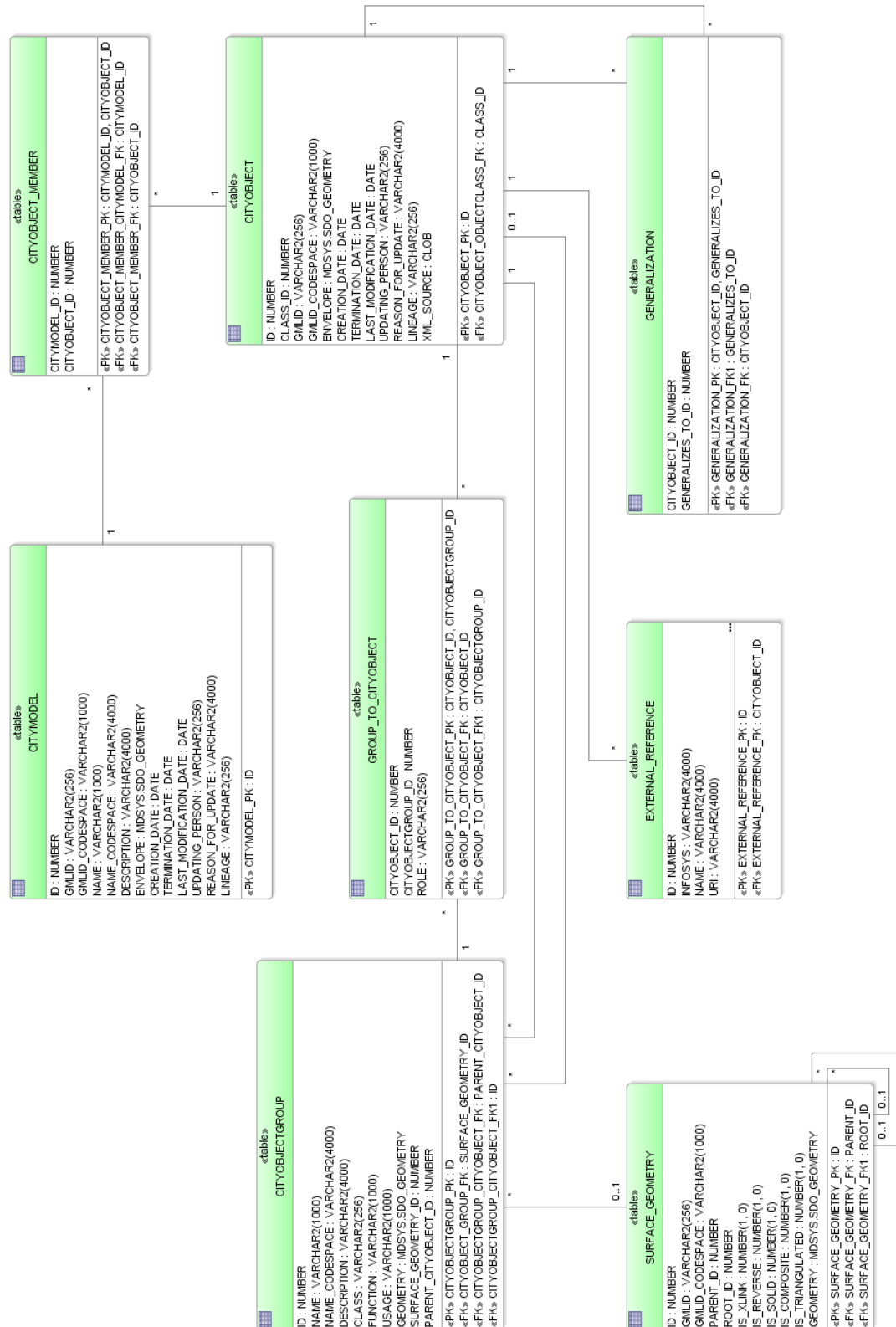


Figure 19: Database schema of the CityGML core elements

2.3.2.2 Tables for geometry representation

The representation of the geometry stored in table `SURFACE_GEOMETRY` differs substantially from the UML chart explained in the CityGML specification; nevertheless it offers about the same functionality:

`SURFACE_GEOMETRY`, `SURFACE_GEOMETRY_SEQ`

In the DB schema the geometry consists of planar surfaces which correspond each to one entry in the table `SURFACE_GEOMETRY`. The geometry is stored as attribute `GEOMETRY` of type `SDO_GEOMETRY` (in each case exactly one planar polygon, possibly including holes). Any surface may have textures or a colour on both sides. Textures are stored within the tables which implement the appearance model (cf. chapter 2.3.2.3).

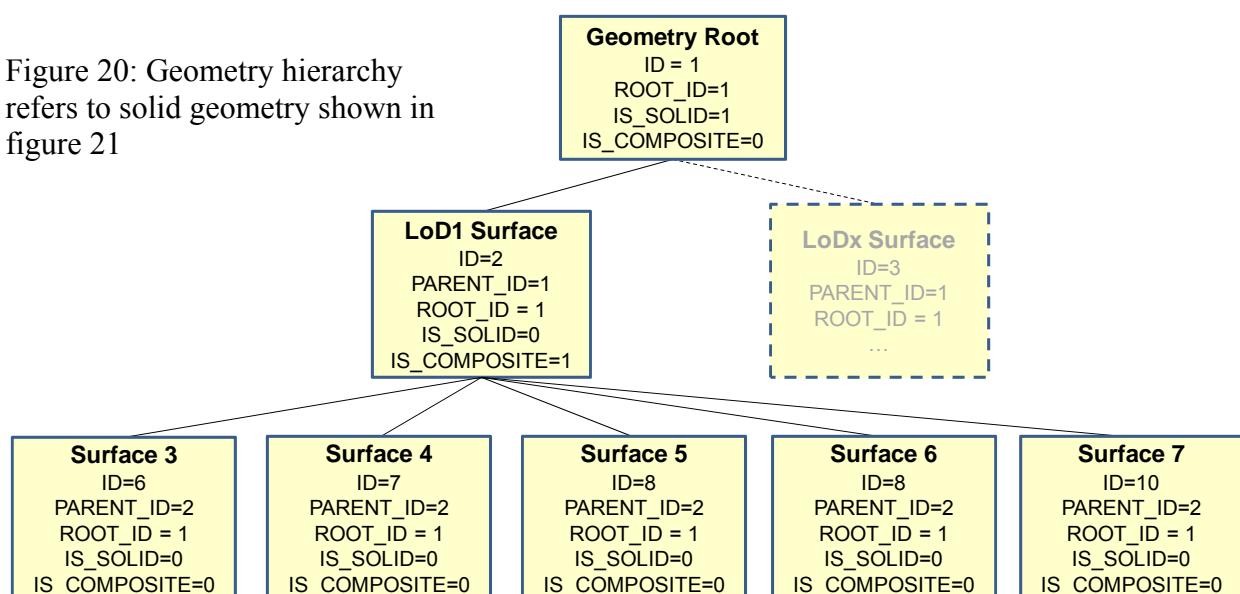
The `SDO_GEOMETRY` in the field `GEOMETRY` of the table `SURFACE_GEOMETRY` is limited as follows:

- the `SDO_GTYPE` must have the type *Polygon*, i.e. a polygon with 3D coordinates (`SDO_GTYPE` = 3003), and
- the `SDO_ETYPE` must be 1003/2003 with `SDO_INTERPRETATION` = 1 (i.e. polygon with 3D coordinates in the boundary, bounded just by line segments, possibly including holes).
- In addition Oracle allows the representation of a rectangle by two corner points (`SDO_ETYPE`=1003/2003, with `SDO_INTERPRETATION` = 3).

A solid is the basis for 3-dimensional geometry. The extent of a solid is defined by the boundary surfaces (outer shell). A shell is represented by a composite surface, where every shell is used to represent a single connected component of the boundary of a solid. It consists of a composite surface (a list of orientable surfaces) connected in a topological cycle. Unlike a ring, a shell's elements have no natural sort order. Like rings, shells are simple.

Surfaces can be aggregated to form a complex of surfaces or the boundary of a volumetric object. The aggregation of multiple surfaces, e.g. F_1 to F_n , (IDs 6 to 10 in figures 20 / 21) is realized the way that the newly created surface tuple F_{n+1} (ID 2) is not assigned a geometry (cf. table 3). Instead, the `PARENT_ID` of the surfaces F_1 to F_n refer to the ID of F_{n+1} .

Figure 20: Geometry hierarchy refers to solid geometry shown in figure 21



In addition, a further tuple (ID 1) is introduced, which represent the solid and defines the root element of the whole aggregation structure. Each surface references to its root, using the `ROOT_ID` attribute. This information has big influence on the system performance, as it

allows to avoid recursive queries. If e.g. the retrieval of all surface elements forming a specific building is of importance, simply those tuples have to be selected which contain the related `ROOT_ID`. On the downside there also follows the limitation that each tuple in `SURFACE_GEOMETRY` can only belong to one aggregate.

Various flags characterise the type of aggregation: `IS_TRIANGULATED` denotes a `TriangulatedSurface`, `IS_SOLID` distinguishes between surface (0) and solid (1), and `IS_COMPOSITE` defines whether this is an aggregate (e.g., `MultiSolid`, `MultiSurface`) or a composite (e.g., `CompositeSolid`, `CompositeSurface`).

Based on these flags the geometry types listed in Table 2 can be distinguished. To distinguish a `MultiSolid` from a `MultiSurface` its child elements have to be analysed: In case the child is a `Solid`, the geometry can be identified as `MultiSolid`.

	isSolid	isComposite	isTriangulated	Geometry
Polygon, Triangle, Rectangle	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	SDO_GEOMETRY(...)
MultiSurface	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
CompositeSurface	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL
TriangulatedSurface	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	NULL
Solid	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
MultiSolid	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	NULL
CompositeSolid	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	NULL

Table 2: Attributes determining aggregation types

Aggregated surfaces can be grouped again with other (compound) surfaces, by generating a common parent. This way, arbitrary aggregations of `Surfaces`, `CompositeSurfaces`, `Solids`, `CompositeSolids` can be formed. Since all tuples in an aggregated geometry refer to the same `ROOT_ID` all tuples can be retrieved efficiently from the table by selecting those tuples with the same `ROOT_ID`.

The aggregation schema allows for the definition of nested aggregations (hierarchy of components). For example, a building geometry (*CompositeSolid*) can be composed of the house geometry (*CompositeSolid*) and the garage geometry (*Solid*), while the house's geometry is further decomposed into the roof geometry (*Solid*) and the geometry of the house body (*Solid*).

In order to provide a unique identifier in table `SURFACE_GEOMETRY`, the next available ID value is provided by the sequence `SURFACE_GEOMETRY_SEQ`.

Example: The geometry shown in the figure below consists of seven surfaces which delimit a volumetric object. In the table it is represented by the following rows:

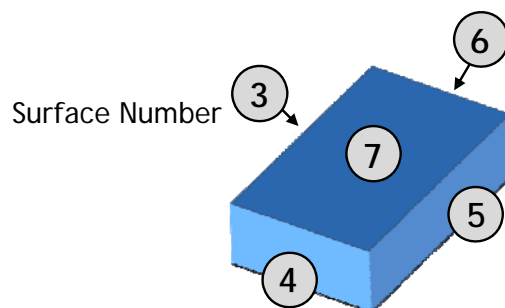


Figure 21: LoD 1 building - closed volume bounded by a `CompositeSurface` which consists of single polygons

SURFACE_GEOMETRY						
ID	GMLID	PARENT_ID	ROOT_ID	IS_SOLID	IS_COMPOSITE	GEOMETRY
1	UUID_Iod1	NULL	1	1	0	NULL
2	Iod1Surface	1	1	0	1	NULL
6	Left1	2	1	0	0	SDO_GEOMETRY for surface 3
7	Front1	2	1	0	0	SDO_GEOMETRY for surface 4
8	Right1	2	1	0	0	SDO_GEOMETRY for surface 5
9	Back1	2	1	0	0	SDO_GEOMETRY for surface 6
10	Roof1	2	1	0	0	SDO_GEOMETRY for surface 7

Table 3: Excerpt of table SURFACE_GEOMETRY
representing the example given in Figure 21

In addition, two further attributes are included in SURFACE_GEOMETRY: IS_XLINK and IS_REVERSE.

IS_XLINK

CityGML allows for sharing of geometry objects between different geometries or different thematic features using the XLink concept of GML3. For this purpose, the geometry object to be shared is assigned an unique *gml:id* which may be referenced by a GML geometry property element through its *xlink:href* attribute. This concept allows for avoiding data redundancy. Furthermore, CityGML does not employ the built-in topology package of GML3 but rather uses the XLink concept for the explicit modelling of topology (see Gröger et al. 2008, p. 25).

Although an XLink can be seen as a pointer to an existing geometry object the SURFACE_GEOMETRY table does not offer a foreign key attribute which could be used to refer to another tuple within this table. The main reason for this is that the referenced tuple typically belongs to a different geometry aggregate, e.g. a different *gml:Solid* object, and thus contains different values for its ROOT_ID and PARENT_ID attributes. Therefore, foreign keys would violate the aggregation mechanism of the SURFACE_GEOMETRY table.

The recommended way of resolving of XLink references to geometry objects requires two steps: First, the referenced tuple of the SURFACE_GEOMETRY table has to be identified by searching the GMLID column for the referenced *gml:id* value. Second, all attribute values of the identified tuple have to be copied to a new tuple. However, the ROOT_ID and PARENT_ID of this new tuple have to be set according to the context of the referencing geometry property element.

Please note:

1. If the referenced tuple is the top of an aggregation (sub)hierarchy within the SURFACE_GEOMETRY table then also **all nested tuples have to be recursively copied** and their ROOT_ID and PARENT_ID have to be adapted.
2. Copying existing entries of the SURFACE_GEOMETRY table results in tuples sharing the same GMLID and GMLID_CODESPACE. Thus, these values cannot be used as a primary key.

When it comes to exporting data to a CityGML instance document, XLink references can be rebuilt by keeping track of the GMLID values of exported geometry tuples. Generally, for *each and every* tuple to be exported it has to be checked whether a geometry object with the same GMLID value has already been processed. If so, the export routine should make use of an XLink reference.

However, checking the GMLID of each and every tuple may dramatically slow down the export process. For this reason, the IS_XLINK flag of the SURFACE_GEOMETRY has been introduced. It may be used to explicitly mark just those tuples for which a corresponding check has to be performed. The IS_XLINK flag should be used in the following manner. The Importer/Exporter provides a corresponding reference implementation.

1. During import

- a. By default, the IS_XLINK flag is set to “0”.
- b. If existing tuples have to be copied due to an XLink reference, IS_XLINK has to be set to “1” for *each and every* copy. Please note, that this rule comprises all copies of nested tuples.
- c. Furthermore, IS_XLINK has to be set to “1” on the original tuple addressed by the XLink reference. If this tuple is the top of an aggregation (sub)hierarchy, IS_XLINK remains “0” for all nested tuples.

2. During export

- a. The export process just has to keep track of the GMLID values of those geometry tuples where IS_XLINK is set to “1”.
- b. When it comes to exporting a tuple with IS_XLINK set to “1”, the export process has to check whether it already came across the same GMLID and, thus, can make use of an XLink reference in the instance document.
- c. For each tuple with IS_XLINK=0 no further action has to be taken.

Especially due to (2c), the IS_XLINK attribute helps to significantly speed up the export process when rebuilding XLink references. Please note, that this is the only intended purpose of the IS_XLINK flag.

You will find a short example explaining the IS_XLINK mechanism on page 51.

IS_REVERSE

The IS_REVERSE flag is used in the context of gml:OrientableSurface geometry objects. Generally, an OrientableSurface instance cannot be represented within the SURFACE_GEOMETRY table since it cannot be encoded using the flags IS_SOLID, IS_COMPOSITE, and IS_TRIANGULATED (cf. Table 2). However, the IS_REVERSE flag is used to encode the information provided by an OrientableSurface and to rebuild OrientableSurfaces during data export.

According to GML3, an OrientableSurface consists of a base surface and an orientation. If the orientation is “+”, then the OrientableSurface is identical to the base surface. If the orientation is “-“, then the OrientableSurface is a reference to a surface with an up-normal that reverses the direction for this OrientableSurface.

During import, only the base surfaces are written to the SURFACE_GEOMETRY table. The following rules have to be obeyed in the context of OrientableSurface:

1. If the orientation of the OrientableSurface is “-“, then
 - a. The direction of the base surface has to be reversed prior to importing it (generally, this means reversing the order of coordinate tuples).
 - b. The IS_REVERSE flag has to be set to “1” for the corresponding entry in the SURFACE_GEOMETRY table.
 - c. If the base surface is an aggregate, then steps (a) and (b) have to be recursively applied for all of its surface members.
2. If the OrientableSurface is identical to its base surface (i.e., if its orientation is “+”), then the base surface can be written to the SURFACE_GEOMETRY table without

taking any further action. The IS_REVERSE flag has to be set to “0” (which is also the *default value*).

3. Please note, that it is not sufficient to just rely on the *gml:orientation* attribute of an OrientableSurface in order to determine its orientation since OrientableSurfaces may be arbitrarily nested.

Flipping the direction of the base surface in step (1a) is essential in order to guarantee that the SDO_GEOMETRY objects stored within the GEOMETRY column are always correctly oriented. This enables applications to just access the GEOMETRY column without having to interpret further attributes of the SURFACE_GEOMETRY table. For example, in the case of a viewer application this allows for a fast rendering of a virtual 3d city scene.

When exporting CityGML instance documents, the IS_REVERSE flag can be used to rebuild OrientableSurface in the following way:

1. If the IS_REVERSE flag is set to “1” for a table entry, the exporter routine has to reverse the direction of the corresponding surface object prior to exporting it (again, this means reversing the order of coordinate tuples).
2. The surface object has to be wrapped by a *gml:OrientableSurface* object with *gml:orientation*=”-”.
3. If the surface object is an aggregate, its surface members having the *same value* for the IS_REVERSE flag *may not* be embraced by another OrientableSurface. However, if the IS_REVERSE value changes, e.g., from “1” for the aggregate to “0” for the surface member, also the surface member has to be embraced by a *gml:OrientableSurface* according to (2). Since there might be nested structures of arbitrary depth this third rule has to be applied recursively.

Like with the IS_XLINK flag, the Importer/Exporter tool provides a reference implementation of the IS_REVERSE flag.

APPEARANCE, APPEARANCE_SEQ



Figure 22: Appearance database schema

SURFACE_DATA, APPEAR_TO_SURFACE_DATA

An appearance is composed of data for each surface geometry object. Information on the data types and its appearance are stored in table `SURFACE_DATA`.

`IS_FRONT` determines the side a surface data object applies to (`IS_FRONT=1`: front face `IS_FRONT=0`: back face of a surface data object). The character string `TYPE` denotes if materials or textures are used for the specific object (values: *X3DMaterial*, *Texture* or *GeoreferencedTexture*). Materials are specified by the attributes `X3D_xxx` which define its graphic representation. Attributes `TEX_xxx` describe usage and features of raster-based 2D textures. The link to corresponding images for example is specified by the attribute `TEX_IMAGE_URI`. The texture image can be stored within this table in the attribute `TEX_IMAGE` using the Oracle datatype `ORDSYS.ORDIMAGE`. Details on using georeferenced textures, such as orientation and reference point, are contained in attributes `GT_xxx`. See chapter 2.2.1.2 for more information on `SURFACE_DATA` attributes or the CityGML specification [Gröger et al. 2008, p. 32-40] which explains the texture mapping process in detail.

Table `APPEAR_TO_SURFACE_DATA` represents the interrelationship between appearances and surfaces for both seasons.

TEXTUREPARAM

Attributes for mapping textures to objects (point list or transformation matrix) which are defined by the CityGML classes *_TextureParameterization*, *TexCoordList*, and *TexCoordGen* are stored in the table `TEXTUREPARAM`.

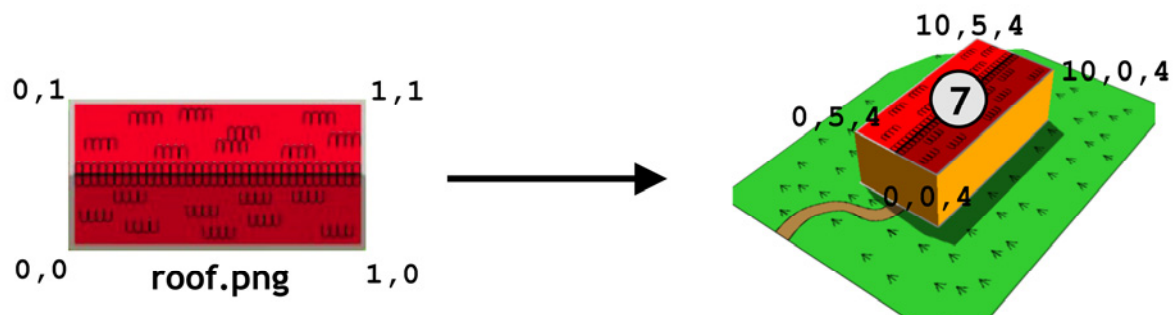


Figure 23: Simple example explaining texture mapping using texture coordinates

TEXTUREPARAM				
SURFACE_GEOMETRY_ID	IS_TEXTURE_PARAMETERIZATION	WORLD_TO_TEXTURE	TEXTURE_COORDINATES	SURFACE_DATA_ID
10	1	NULL	0.0 0.0 1.0 0.0 1.0 1.0 0.0 1.0 0.0 0.0	20
...

Table 4: Example for table `TEXTUREPARAM`

Texture coordinates are applicable to polygonal surfaces, whose boundaries are described by a closed linear ring (last coordinate is equal to first). Coordinates are stored as string with a list of decimal values separated by whitespaces. The `WORLD_TO_TEXTURE` attribute defines a transformation matrix from a location in world space to texture space. For more details see the CityGML Implementation Specification [Gröger et al. 2007, 2008].

SURFACE_DATA (excerpt)					
ID	IS_FRONT	TYPE	X3D_DIFFUSE_COLOR	TEX_IMAGE_URI	TEX_WRAP_MODE
20	1	ParameterizedTexture	NULL	roof.png	none
...					

Table 5: Example showing an excerpt of SURFACE_DATA table

The following more detailed example should clarify the interaction between the tables needed to handle the appearance representation. For a specific building two levels of detail (LoD1 and LoD2) are given with two appearances for summer and winter season (cf. Figure 24). The table APPEARANCE contains information about the themes available for the specific building: summer (ID=1) and winter theme (ID=2).

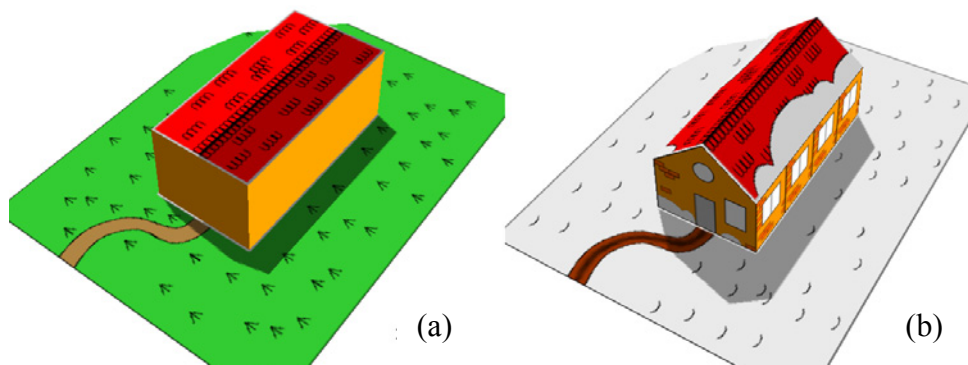


Figure 24: Visualisation of a simple building in LoD1 and LoD2 using the appearance model. Two themes are defined for the building and the surrounding terrain: (a) building in summertime and (b) building in wintertime

Six surface representations are listed in table SURFACE_DATA (cf. table 7). First of all, a homogeneous material is defined (ID=1), represented by a 3-component (RGB) colour value which will be used for both appearances (summer and winter). This also applies to a general side façade texture (ID=3, Figure 25 right) which is repeated (wrapped) to fill the entire surface. For each of the front side, the back side and the ground two images are available: parameterized ones for the sides (Figure 25 left and middle) and georeferenced ones for the ground and the roof surfaces (figure 27). The attribute TEX_IMAGE_URI contains the information which is needed to locate the texture files. The texture may even be stored within the table in the column TEX_IMAGE (data type must be ORD.IMAGE). The coordinates for mapping the textures to the object are stored in table TEXTUREPARAM. For the general side texture (SURFACE_DATA_ID=3) five coordinate pairs are needed to define a closed ring (here: rectangle). Table SURFACE_GEOMETRY contains the information of all geometry parts that form the building and its appropriate 3D coordinates (cf. tables on page 48).

An important feature of the database is related to the storage of texture coordinates of inner rings, which are not supported explicitly. This can be handled indirectly with help of attribute TEXTURE_COORDINATES (Type: VARCHAR2) of table TEXTUREPARAM: For every ring its texture coordinates are stored as string of double values separated by space. If a face is described by multiple rings its texture coordinate strings are separated by semicolon where the sequence is of importance. At first the coordinates of the outer ring are stored followed by the list of the inner rings, which has to match the sequence of the corresponding

MDSYS.SDO_GEOMETRY objects saved in the SURFACE_GEOMETRY table. If texture coordinates are not available describing the inner ring a warning message is displayed during the import process.

See the following page for an example of the storage of appearances in the city database. Figures 25 and 27 show the images used for texturing a building in LoD2. In LoD1, a material definition is used to define the wall colors of the building. Table 6 to Table 8 show a combination of tables representing the building's textures. There are different images available for summer and winter resulting in two themes: Summer and Winter. The tuples within the tables are color-coded according to their relation to the respective theme:

- Green: only summer related data
- Light-gray: only winter related data
- Orange: both summer and winter related data

Figure 26 shows the LoD2 representation of summer appearances (theme Summer).

APPEARANCE					
ID	GMLID	GMLID_CODESPACE	THEME	CITYMODEL_ID	CITYOBJECT_ID
...		
1	App1	file:/C:/example.gml	Summer		1000
2	App2	file:/C:/example.gml	Winter		1000
...		

Table 6: Excerpt of table APPEARANCE
The relation to the building feature is given by the foreign key CITYOBJECT_ID

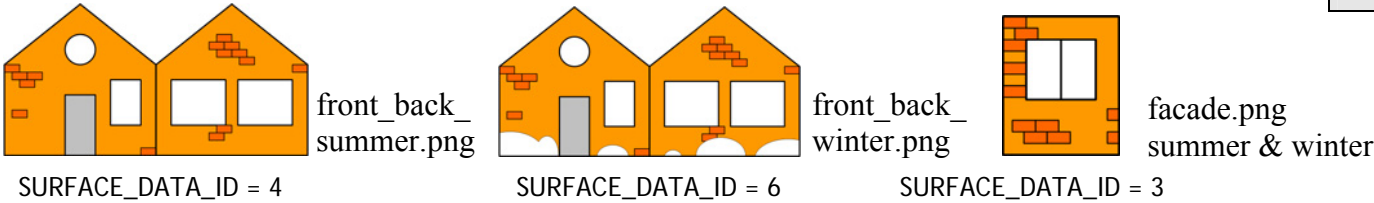


Figure 25: Images for parameterized textures

SURFACE_DATA							
ID	IS_FRONT	TYPE	X3D_DIFFUSE_COLOR	TEX_IMAGE_URI	TEX_WRAP_MODE	GT_ORIENTATION	GT_REFERENCE_POINT
7	1	X3DMaterial	1.0 0.6 0.0				
3	1	ParameterizedTexture		facade.png	wrap		
4	1	ParameterizedTexture		front_back_summer.png	none		
6	1	ParameterizedTexture		front_back_winter.png	none		
8	1	GeoreferencedTexture		ground_summer.png	none	0.05 0.0 0.0 0.066667	SDO_GEOMETRY(...)
5	1	GeoreferencedTexture		ground_winter.png	none	0.05 0.0 0.0 0.066667	SDO_GEOMETRY(...)

Table 7: Excerpt of table SURFACE_DATA

TEXTUREPARAM					COMMENTS
SURFACE_ GEOMETRY_ID	IS_TEXTURE_ PARA-METRIZATION	WORLD_TO_ TEXTURE	TEXTURE_COORDINATES	SURFACE_ DATA_ID	
30	0	NULL	NULL	8	LoD 2 ground S
16	0	NULL	NULL	8	LoD 2 roof left S
17	0	NULL	NULL	8	LoD 2 roof right S
13	1	NULL	0.0 0.0 0.5 0.0 0.5 0.6 0.25 1.0 0.0 0.6 0.0 0.0	4	LoD 2 front S
15	1	NULL	0.5 0.0 1.0 0.0 1.0 0.6 0.75 1.0 0.5 0.6 0.5 0.0	4	LoD 2 back S
12	1	NULL	0.0 0.0 2.0 0.0 2.0 1.0 0.0 1.0 0.0 0.0	3	LoD 2 façade left S/W
11	1	NULL	0.0 0.0 2.0 0.0 2.0 1.0 0.0 1.0 0.0 0.0	3	
14	1	-0.4 0.0 0.0 1.0 0.0 0.0 0.3333 0.0 0.0 0.0 0.0 1.0	NULL	3	LoD 2 façade right S/W
30	0	NULL	NULL	5	LoD2 ground W
16	0	NULL	NULL	5	LoD 2 roof left W
17	0	NULL	NULL	5	LoD 2 roof right W
13	1	NULL	0.0 0.0 0.5 0.0 0.5 0.6 0.25 1.0 0.0 0.6 0.0 0.0	6	LoD 2 front W
15	1	NULL	0.5 0.0 1.0 0.0 1.0 0.6 0.75 1.0 0.5 0.6 0.5 0.0	6	LoD 2 back W
2	0	NULL	NULL	7	LoD1 walls S/W
10	0	NULL	NULL	8	LoD1 roof S/W

Table 8: Table TEXTUREPARAM

APPEAR_TO_SURFACE_DATA		COMMENTS
APPEARANCE_ID	SURFACE_DATA_ID	
1	7	LoD1 S
2	7	LoD1 W
1	8	LoD2 ground/roof S
1	3	LoD2 façade S
1	4	LoD2 front/back S
2	5	LoD2 ground/roof W
2	3	LoD2 façade W
2	6	LoD2 front/back W

Table 6a: APPEAR_TO_SURFACE table

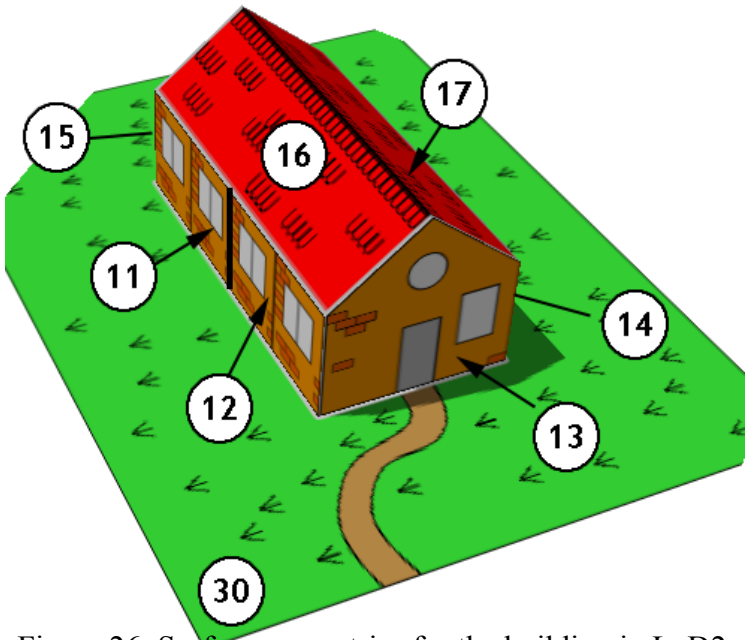
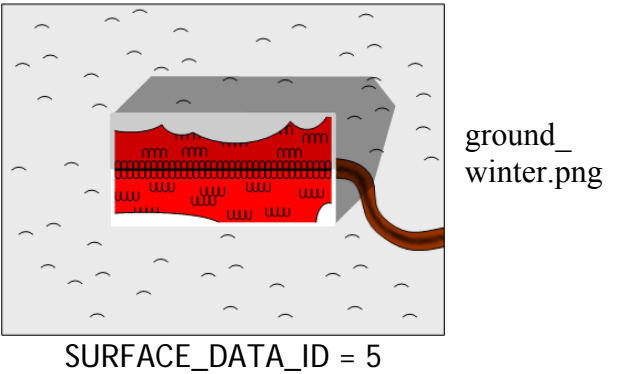
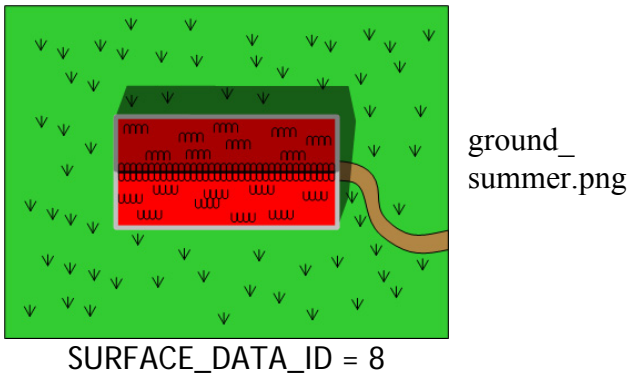


Figure 26: Surface geometries for the building in LoD2 (the IDs for LoD1 are the same as in fig.20)



SURFACE_DATA_ID = 5



SURFACE_DATA_ID = 8

The image *ground_winter.png* is assigned to the terrain and the roof surfaces of the building both in LoD1 and LoD2 within the winter theme (a), *ground_summer.png* within the summer theme (b)

Figure 27: Images for georeferenced textures

Building Model

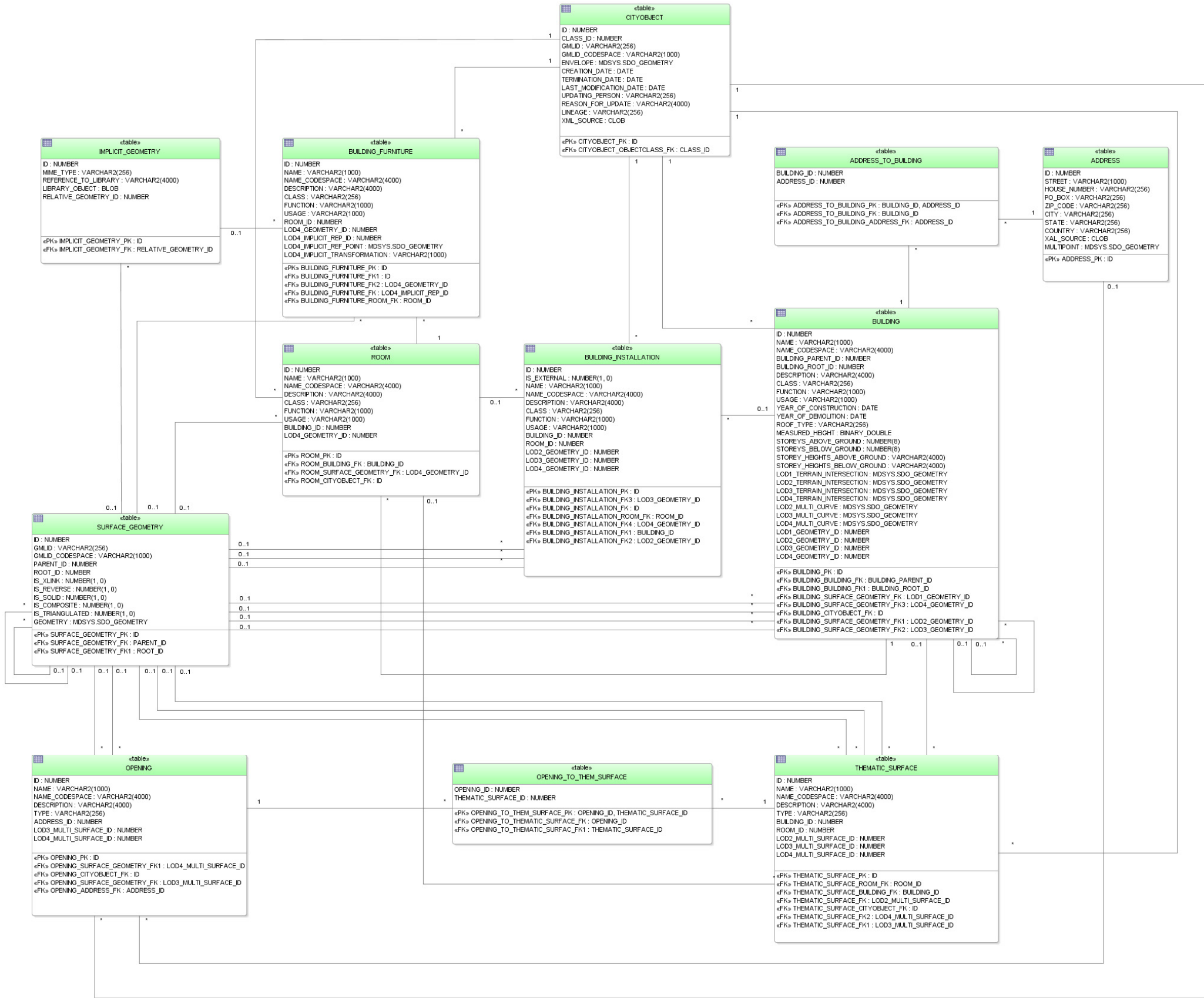


Figure 28: Building database schema

BUILDING

The building model, described in paragraph 2.2.1.3.2 at the conceptual level, is realised by the tables shown in Figure 28. The three CityGML classes *AbstractBuilding*, *Building* and *BuildingPart* are merged into the single table BUILDING. The subclass relationship with CITYOBJECT arises from using identical IDs, i.e. for each tuple in BUILDING there must exist a tuple within CITYOBJECT with the same ID.

The meaning and the name of most fields are identical to those of the attributes in the UML diagram (cf. Figure 6). Geometry is represented by the four foreign keys LOD1_GEOMETRY_ID to LOD4_GEOMETRY_ID which refer to entries in the SURFACE_GEOMETRY table and represent each LoD's geometry.

The component hierarchy within a building is realized by the foreign key BUILDING_PARENT_ID which refers to the superordinate building (aggregate) and contains NULL, if such does not exist. This way, a tree-like structure arises also for building aggregates. BUILDING_PARENT_ID points at the predecessor in the tree. The foreign key BUILDING_ROOT_ID refers directly to the top level (root) of a building tree. In order to select all parts forming a building one only has to select those with the same BUILDING_ROOT_ID.

Optionally the geometry of the terrain intersection curve is stored in the attribute LODx_TERRAIN_INTERSECTION ($1 \leq x \leq 4$) as Oracle SDO_GEOMETRY (MultiCurve). Additional line-typed building elements such as antennas are optionally modelled by the attribute LODx_MULTI_CURVE ($1 \leq x \leq 4$, also using SDO_GEOMETRY (MultiCurve)).

OPENING

Openings (CityGML class *Opening*) are represented by the table OPENING. No individual tables are created for the subclasses. Instead, the differentiation is achieved by the attribute TYPE in OPENING. Valid values are the Strings 'Door' and 'Window'. Its spelling must exactly fit to the CityGML classname (case sensitive!). Table OPENING_TO_THEM_SURFACE associates an opening ID in table OPENING with a thematic surface ID in table THEMATIC_SURFACE representing the m:n relation between both tables. An address can be assigned to a door (table OPENING) by the foreign key ADDRESS_ID in the table OPENING. Furthermore, addresses may be assigned to buildings (see table ADDRESS for detailed information).

THEMATIC_SURFACE

The table THEMATIC_SURFACE represents thematic boundary features. CityGML class *BoundarySurface* has a number of concrete subclasses representing different types of surfaces. One possibility would be to represent each of these classes by its own table. Here, we choose the approach to create one table representing all those classes. No own tables for the subclasses of BoundarySurface were created in the table schema; instead, the type of the boundary surface is given by the attribute TYPE of table THEMATIC_SURFACE. Allowed values are the names of the subclasses of *BoundarySurface* ("ClosureSurface", "GroundSurface", "WallSurface", "RoofSurface", "FloorSurface", "InteriorWallSurface", "CeilingSurface").

The aggregation relation between buildings and the corresponding boundary surfaces results from the foreign key BUILDING_ID of the table THEMATIC_SURFACE which refers to the ID of the respective building. Thematic surfaces and the corresponding building should share their geometry: the geometry should be defined only once and be used conjointly as XLinks.

The `SURFACE_GEOMETRY`, which for example geometrically defines a roof, should at the same time be a part of the volume geometry of the building the roof belongs to.

Example:

In Figure 29 a building geometry is shown which consists of several MultiSurfaces. Please notice, that the left wall (ID 5) is composed of two polygons (IDs 11 and 12) and the roof is splitted into a left and right part (IDs 20 and 21) each of them formed by two parts, the roof surface and an overhang. In table `SURFACE_GEOMETRY` (cf. Table 9) the value 1 is set for attribute `IS_COMPOSITE` characterising tuples with IDs 5, 20 and 21 as combined surfaces. For adding semantic information table `THEMATIC_SURFACE` is needed, which contains a tuple that references ID 30 in table `SURFACE_GEOMETRY`. The attribute `TYPE` identifies the related tuple (ID 30) as `RoofSurface` (cf. Table 11). As can be seen in table 9, the `IS_XLINK` mechanism is used in table `SURFACE_GEOMETRY`. Copies of the left roof (ID 21) and its related parts (IDs 16 and 18) are stored as new tuples (IDs 31 to 33) and are marked by the `IS_XLINK` flag, as well as the original tuple (ID 21).

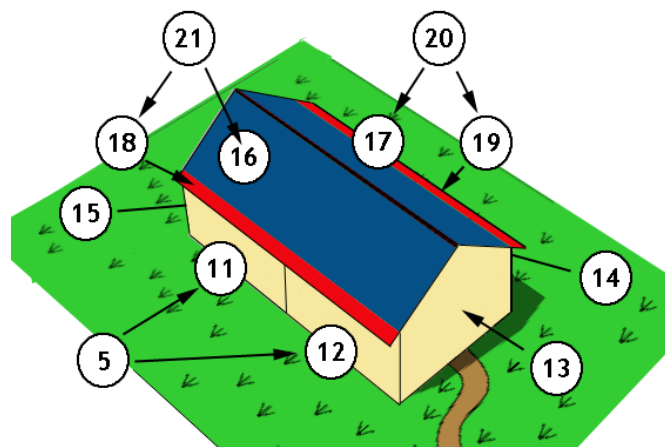


Figure 29: LoD2 building with roof overhangs, highlighted in red.

SURFACE_GEOMETRY (excerpt)							
ID	GMLID	PARENT_ID	ROOT_ID	IS_SOLID	IS_COMPOSITE	IS_XLINK	GEOMETRY
3	UUID_LoD2	NULL	3	0	0	0	NULL
5	Left_Wall	3	3	0	1	0	NULL
11	Left_Wall_1	5	3	0	0	0	Geometry comp (5-1) surface 11
12	Left_Wall_2	5	3	0	0	0	Geometry comp (5-2) surface 12
13	Front	3	3	0	0	0	Geometry surface 13
14	Right_Wall	3	3	0	0	0	Geometry surface 14
15	Back	3	3	0	0	0	Geometry surface 15
16	Roof_part_1	21	3	0	0	0	Geometry surface 16
17	Roof_part_2	20	3	0	0	0	Geometry surface 17
18	Overhang_1	21	3	0	0	0	Geometry of overhang 18
19	Overhang_2	20	3	0	0	0	Geometry of overhang 19
20	Roof_right	3	3	0	1	0	NULL
21	Roof_left	3	3	0	1	1	NULL
30	ROOF_SURFACE	NULL	30	0	0	0	NULL
31	Roof_left	30	30	0	1	1	NULL
32	Roof_part_1	31	30	0	0	1	Geometry surface 16
33	Overhang_1	31	30	0	0	1	Geometry of overhang 18
...

Table 9: Excerpt of table `SURFACE_GEOMETRY`
Geometry objects are stored as Oracle datatype `MDSYS.SDO_GEOMETRY`

BUILDING					
ID	BUILDING_ROOT_ID	...	LOD1_GEOMETRY_ID	LOD2_GEOMETRY_ID
...		
1	1	1	3
...		

Table 10: Excerpt of table BUILDING

THEMATIC_SURFACE						
ID	TYPE	BUILDING_ID	ROOM_ID	LOD2_MULTI_SURFACE_ID	...
...
70	RoofSurface	1	NULL	30
...

Table 11: Excerpt of table THEMATIC_SURFACE

BUILDING_INSTALLATION

The UML classes BuildingInstallation and IntBuildingInstallation are realized by the single table BUILDING_INSTALLATION. Internal and external objects are distinguished by the attribute IS_EXTERNAL (external 1, internal 0). The relation to the corresponding building arises from the foreign key BUILDING_ID, whereas the geometry in LoD 2 to 4 is given via the foreign keys LODx_GEOMETRY_ID ($2 \leq x \leq 4$) referring to the table SURFACE_GEOMETRY.

ROOM

Room objects are allowed in LoD4 only. Therefore the only key LOD4_GEOMETRY_ID is referring to the table SURFACE_GEOMETRY. Additionally the foreign keys to tables BUILDING and CITYOBJECT are necessary to map the relationship to these tables.

BUILDING_FURNITURE

As rooms may be equipped with furniture (chairs, wardrobes, etc.), a foreign key referencing to ROOM_ID is mandatory. The geometry of furniture objects can be described explicitly using the foreign key LOD4_GEOMETRY_ID or by using prototypes which are stores stored as library objects. The information needed for mapping prototype objects to rooms consists of a base point and a transformation matrix stored in the attributes LOD4_IMPLICIT_REF_POINT and LOD4_IMPLICIT_TRANSFORMATION.

ADDRESS, ADDRESS_TO_BUILDING, and ADDRESS_SEQ

Addresses are realized by the table ADDRESS. The m:n relation with buildings arises from the table ADDRESS_TO_BUILDING which associates a building ID and an address ID. An address can also be assigned to a door (table OPENING) by the foreign key ADDRESS_ID in the table OPENING..

The next available ID for the table ADDRESS is provided by the sequence ADDRESS_SEQ.

2.3.2.4 CityFurniture Model

CITY_FURNITURE

Attributes of the class CityFurniture specified in the UML (cf. figure 8) diagram are directly mapped onto the tables (*class* and *function*).

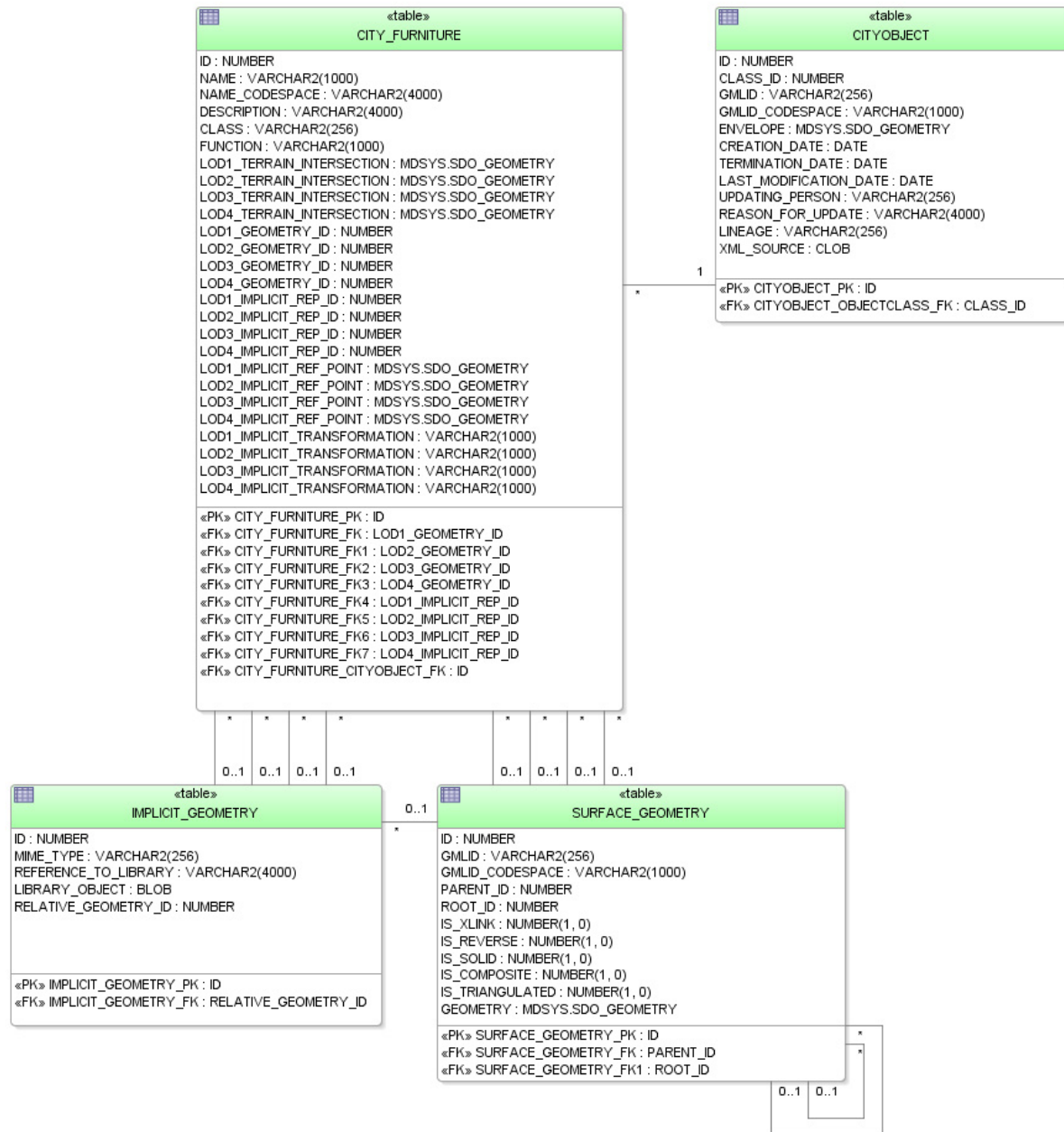


Figure 30: CityFurniture database schema

The geometry of city furniture objects is represented either as a geometry object (LOD x GEOMETRY_ID, where $1 \leq x \leq 4$) related to table SURFACE_GEOMETRY or as implicit geometry. In the case of an ImplicitGeometry, a reference point of the object (LOD x _IMPLICIT_REF_POINT, with $1 \leq x \leq 4$) and optionally a transformation matrix (LOD x _IMPLICIT_TRANSFORMATION, with $1 \leq x \leq 4$) must be given. In order to compute the actual location of the object, the transformation of the local coordinates into the reference system of the city model must be processed and the anchor point coordinates (LOD x _IMPLICIT_REF_ID, with $1 \leq x \leq 4$) must be added (cf Gröger et al 2008 p. 26).

2.3.2.5 Digital Terrain Model

RELIEF

A tuple in the table RELIEF represents a complex relief object, which consists of different relief components. It has an attribute LODGROUP that describes the affiliation of the relief object to a certain level of detail (LoD) of the city model. In addition, it comprises the attribute NAME that contains the name of the relief. The individual components of a complex relief object are stored in the tables BREAKLINE_RELIEF, MASSPOINT_RELIEF, TIN_RELIEF and RASTER_RELIEF. Every relief component has an attribute LoD that describes the affiliation to a certain level of detail (resolution, accuracy). However, individual components of a complex relief object may belong to different LoD and be heterogeneous, i.e. a mixture of TINs, grids and mass points. The geometrical separation between the individual relief components of a complex relief object is realized via polygons (attribute EXTENT), which specify the validity area of the relief component. Every relief component has an attribute NAME that is used for naming of the component. The relief as well as every relief component are derived from CITYOBJECT and receive the same ID as the CityObject. Table RELIEF_FEAT_TO_REL_COMP represents the interrelationship between relief features and relief components.

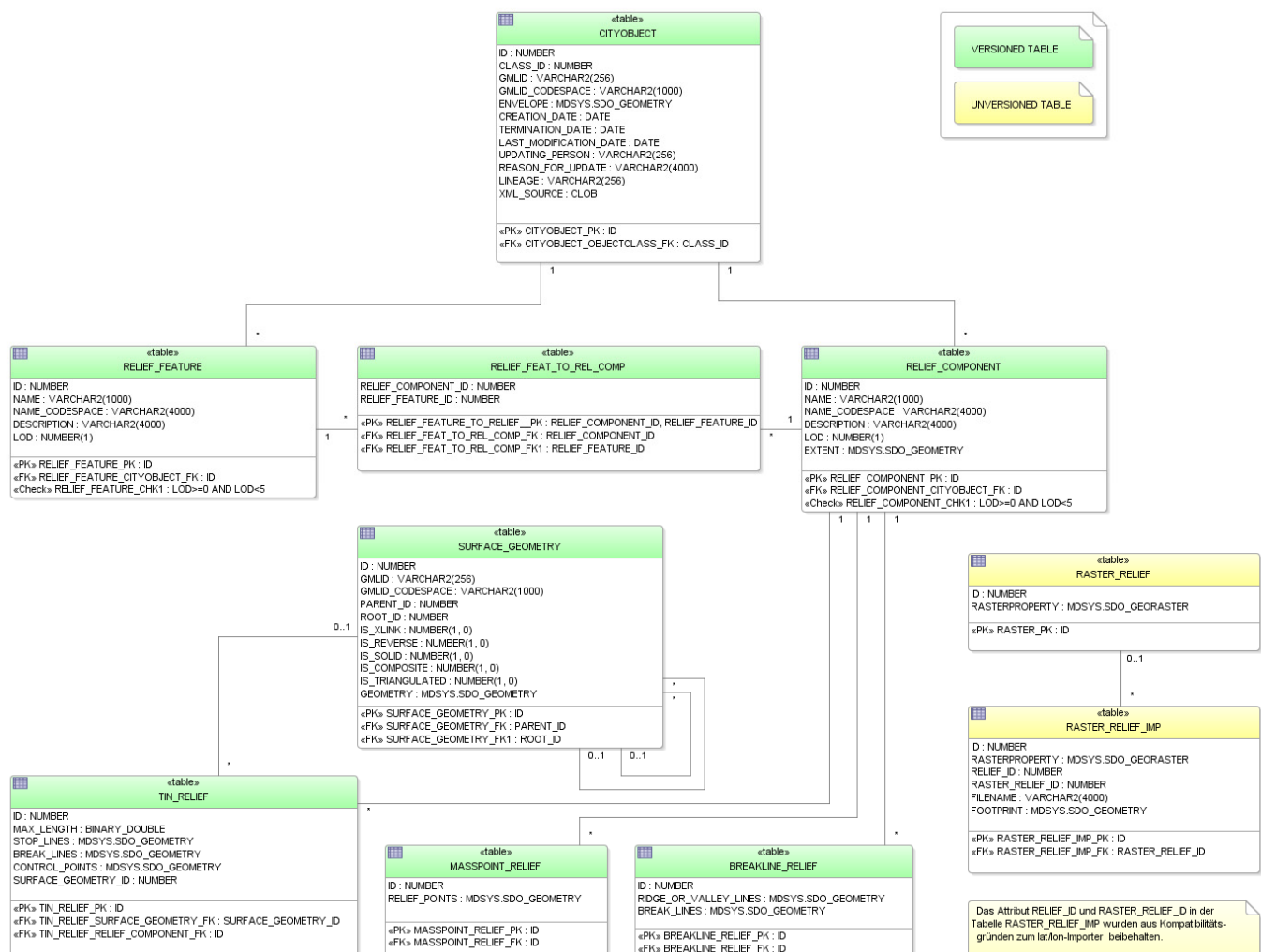


Figure 31: Digital Terrain Model database schema

The tables RASTER_RELIEF and RASTER_RELIEF_IMP cannot be versioned using the Oracle Workspace Manager.

Geometry attributes of the Oracle data type SDO_GEOMETRY for different relief components are limited to the following value domains:

- Table BREAKLINE_RELIEF – attributes BREAK_LINES and RIDGE_OR_VALLEY_LINES
SDO_GTYPE: MULTILINE (code defined by Oracle Spatial - 3006)
- Table TIN_RELIEF – attributes STOP_LINES, BREAK_LINES,
SDO_GTYPE: MULTILINE (3006)
attribute RELIEF_POINTS
SDO_GTYPE: MULTIPOINT (3001 or 3005)
alternatively TIN triangles could be stored as triangulated surfaces in table SURFACE_GEOMETRY
- Table MASSPOINT_RELIEF – attribute RELIEF_POINTS
SDO_GTYPE: MULTIPOINT (3001 or 3005)

The attribute EXTENT in table RELIEF_COMPONENT, which defines the validity extents of each relief component, is represented by a footprint polygon of datatype

- SDO_GTYPE: 3D-POLYGON, i.e. the SDO_GEOMETRY is limited to
SDO_GTYPE=3003, SDO_ETYPE=1003 and SDO_INTERPRETATION=1 or
SDO_INTERPRETATION = 3 for a RECTANGLE.

2.3.2.6 Generic CityObject Model

3D city models will most likely contain attributes, which are not explicitly modelled in CityGML. Moreover, there may be 3D objects that are not covered by the thematic classes of CityGML. Generic objects and attributes help to support the storage of such data.

GENERIC_CITYOBJECT

The geometrical information of a generic cityobject can either be stored using absolute world coordinates as surface geometry (LOD_xGEOMETRY_ID, with $0 \leq x \leq 4$) or as implicit geometry. In the case of an ImplicitGeometry, a reference point of the object (LOD_x_IMPLICIT_REF_POINT, with $0 \leq x \leq 4$) and optionally a transformation matrix (LOD_x_IMPLICIT_TRANSFORMATION, with $0 \leq x \leq 4$) must be given. In order to compute the actual location of the object, the transformation of the local coordinates into the reference system of the city model must be processed and the anchor point coordinates (LOD_x_IMPLICIT_REP_ID, with $0 \leq x \leq 4$) must be added (cf Gröger et al 2008 p. 26).

Moreover, in order to specify the exact intersection of the DTM with the 3D geometry of a *GenericCityObject*, the *TerrainIntersectionCurve* can be added for every LOD (LOD_x_TERRAIN_INTERSECTION, with $0 \leq x \leq 4$ - SDO_GTYPE: MULTILINE (3006)).

CITYOBJECT_GENERICATTRIB, CITYOBJECT_GENERICATT_SEQ

The table CITYOBJECT_GENERICATTRIB is used to represent the concept of generic attributes. However, the creation of a table for every type of attribute was omitted. Instead a single table CITYOBJECT_GENERICATTRIB represents all types and the types are differentiated via the values of the attribute DATATYPE.

The table provides fields for every data type, but only one of those fields is relevant in each case. An overview of the meaning of the entries in the field DATATYPE is given in Table 12.

```

    erDiagram
        OBJECTCLASS ||--o{ CITYOBJECT : "has"
        OBJECTCLASS ||--o{ CITYOBJECT_GENERICATTRIB : "has"
        CITYOBJECT ||--o{ CITYOBJECT_GENERICATTRIB : "has"
        CITYOBJECT ||--o{ SURFACE_GEOMETRY : "has"
        CITYOBJECT_GENERICATTRIB ||--o{ SURFACE_GEOMETRY : "has"

        OBJECTCLASS {
            NUMBER ID PK
            VARCHAR2(256) CLASSNAME
            NUMBER SUPERCLASS_ID
            OBJECTCLASS_PK_1 ID FK
            OBJECTCLASS_OBJECTCLASS_FK_1 SUPERCLASS_ID FK
        }

        CITYOBJECT {
            NUMBER ID PK
            NUMBER CLASS_ID
            VARCHAR2(256) GMLID
            VARCHAR2(1000) GMLID_CODESPACE
            MDSYS.SDO_GEOMETRY ENVELOPE
            DATE CREATION_DATE
            DATE TERMINATION_DATE
            DATE LAST_MODIFICATION_DATE
            VARCHAR2(256) UPDATING_PERSON
            VARCHAR2(4000) REASON_FOR_UPDATE
            VARCHAR2(256) LINEAGE
            CLOB XML_SOURCE
            CITYOBJECT_PK_1 ID FK
            CITYOBJECT_OBJECTCLASS_FK_1 CLASS_ID FK
        }

        GENERIC_CITYOBJECT {
            NUMBER ID PK
            VARCHAR2(1000) NAME
            VARCHAR2(4000) NAME_CODESPACE
            VARCHAR2(4000) DESCRIPTION
            VARCHAR2(256) CLASS
            VARCHAR2(1000) FUNCTION
            VARCHAR2(1000) USAGE
            MDSYS.SDO_GEOMETRY LOD0_TERRAIN_INTERSECTION
            MDSYS.SDO_GEOMETRY LOD1_TERRAIN_INTERSECTION
            MDSYS.SDO_GEOMETRY LOD2_TERRAIN_INTERSECTION
            MDSYS.SDO_GEOMETRY LOD3_TERRAIN_INTERSECTION
            MDSYS.SDO_GEOMETRY LOD4_TERRAIN_INTERSECTION
            NUMBER LOD0_GEOMETRY_ID
            NUMBER LOD1_GEOMETRY_ID
            NUMBER LOD2_GEOMETRY_ID
            NUMBER LOD3_GEOMETRY_ID
            NUMBER LOD4_GEOMETRY_ID
            NUMBER LOD0_IMPLICIT_REP_ID
            NUMBER LOD1_IMPLICIT_REP_ID
            NUMBER LOD2_IMPLICIT_REP_ID
            NUMBER LOD3_IMPLICIT_REP_ID
            NUMBER LOD4_IMPLICIT_REP_ID
            MDSYS.SDO_GEOMETRY LOD0_IMPLICIT_REF_POINT
            MDSYS.SDO_GEOMETRY LOD1_IMPLICIT_REF_POINT
            MDSYS.SDO_GEOMETRY LOD2_IMPLICIT_REF_POINT
            MDSYS.SDO_GEOMETRY LOD3_IMPLICIT_REF_POINT
            MDSYS.SDO_GEOMETRY LOD4_IMPLICIT_REF_POINT
            VARCHAR2(1000) LOD0_IMPLICIT_TRANSFORMATION
            VARCHAR2(1000) LOD1_IMPLICIT_TRANSFORMATION
            VARCHAR2(1000) LOD2_IMPLICIT_TRANSFORMATION
            VARCHAR2(1000) LOD3_IMPLICIT_TRANSFORMATION
            VARCHAR2(1000) LOD4_IMPLICIT_TRANSFORMATION
            GENERIC_CITYOBJECT_PK_1 ID FK
            GENERIC_CITYOBJECT_FK_1 LOD0_IMPLICIT_REP_ID FK
            GENERIC_CITYOBJECT_FK_2 LOD1_IMPLICIT_REP_ID FK
            GENERIC_CITYOBJECT_FK_3 LOD2_IMPLICIT_REP_ID FK
            GENERIC_CITYOBJECT_FK_4 LOD3_IMPLICIT_REP_ID FK
            GENERIC_CITYOBJECT_FK_5 LOD4_IMPLICIT_REP_ID FK
            GENERIC_CITYOBJECT_FK_6 LOD1_GEOMETRY_ID FK
            GENERIC_CITYOBJECT_FK_7 LOD2_GEOMETRY_ID FK
            GENERIC_CITYOBJECT_FK_8 LOD3_GEOMETRY_ID FK
            GENERIC_CITYOBJECT_FK_9 LOD4_GEOMETRY_ID FK
            GENERIC_CITYOBJECT_FK_10 ID FK
        }

        CITYOBJECT_GENERICATTRIB {
            NUMBER ID PK
            VARCHAR2(256) ATTRNAME
            NUMBER(1) DATATYPE
            VARCHAR2(4000) STRVAL
            NUMBER INTVAL
            NUMBER REALVAL
            VARCHAR2(4000) URIVAL
            DATE DATEVAL
            MDSYS.SDO_GEOMETRY GEOMVAL
            BLOB BLOBVAL
            NUMBER CITYOBJECT_ID
            NUMBER SURFACE_GEOMETRY_ID
            CITYOBJECT_GENERICATTRIB_PK_1 ID FK
            CITYOBJECT_GENERICATTRIB_FK_1 CITYOBJECT_ID FK
            CITYOBJECT_GENERICATTRIB_FK_1_1 SURFACE_GEOMETRY_ID FK
        }

        SURFACE_GEOMETRY {
            NUMBER ID PK
            VARCHAR2(256) GMLID
            VARCHAR2(1000) GMLID_CODESPACE
            NUMBER PARENT_ID
            NUMBER ROOT_ID
            NUMBER(1, 0) IS_XLINK
            NUMBER(1, 0) IS_REVERSE
            NUMBER(1, 0) IS_SOLID
            NUMBER(1, 0) IS_COMPOSITE
            NUMBER(1, 0) IS_TRIANGULATED
            MDSYS.SDO_GEOMETRY GEOMETRY
            SURFACE_GEOMETRY_PK_1 ID FK
            SURFACE_GEOMETRY_FK_1 PARENT_ID FK
            SURFACE_GEOMETRY_FK_1_1 ROOT_ID FK
        }

        IMPLICIT_GEOMETRY {
            VARCHAR2(256) TYPE
            VARCHAR2(4000) REFERENCE_TO_LIBRARY
            BLOB SURFACE_OBJECT
            NUMBER SURFACE_GEOMETRY_ID
            IMPLICIT_GEOMETRY_PK_1 ID FK
            IMPLICIT_GEOMETRY_FK_1 RELATIVE_GEOMETRY_ID FK
        }
  
```

The relation between the generic attribute and the corresponding CityObject is established by the foreign key CITYOBJECT ID (REL CITYOBJ ID ID 1).

DATATYPE	attribute type
1	STRING
2	INTEGER
3	REAL
4	URI
5	DATE
6	BLOB
7	Oracle geometry (SDO_GEOMETRY)
8	Geometry via surfaces in the table SURFACE_GEOMETRY

Table 12: Attribute type

OBJECTCLASS

In the (meta)table OBJECTCLASS all class names (attribute CLASSNAME) of the schema are listed. The relation of the subclass to its parent class is represented via the attribute SUPERCLASS_ID in the subclass as a foreign key to the ID of the parent class.

ID	CLASSNAME	SUPERCLASS_ID
0	Undefined	
1	Object	
2	_AbstractFeature	1
3	_CityObject	2
4	LandUse	3
5	GenericCityObject	3
6	_VegetationObject	3
7	SolitaryVegetationObject	6
8	PlantCover	6
9	WaterBody	3
10	_WaterBoundarySurface	3
11	WaterSurface	10
12	WaterGroundSurface	10
13	WaterClosureSurface	10
14	ReliefFeature	3
15	_ReliefComponent	3
16	TINRelief	15
17	MassPointRelief	15
18	BreaklineRelief	15
19	Raster	15
20	Orthophoto	3
21	CityFurniture	3
22	_TransportationObject	3
23	CityObjectGroup	3
24	_AbstractBuilding	3
25	BuildingPart	24
26	Building	24
27	BuildingInstallation	3
28	IntBuildingInstallation	3
29	_BoundarySurface	3
30	CeilingSurface	29
31	InteriorWallSurface	29
32	FloorSurface	29
33	RoofSurface	29
34	WallSurface	29
35	GroundSurface	29
36	ClosureSurface	29
37	_Opening	3
38	Window	37
39	Door	37
40	BuildingFurniture	3
41	Room	3
42	TransportationComplex	22
43	Track	42
44	Railway	42
45	Road	42
46	Square	42
47	TrafficArea	22
48	AuxiliaryTrafficArea	22
49	FeatureCollection	2
50	Appearance	2
51	_SurfaceData	2
52	_AbstractTexture	51
53	X3DMaterial	51
54	ParameterizedTexture	52
55	GeoreferencedTexture	52
56	TextureParametrization	1
57	CityModel	49
58	Address	2
59	ImplicitGeometry	1

Table 13: Class names

2.3.2.7 LandUse Model

LAND_USE

The table LAND_USE contains all attributes specified in the UML diagram (*class, function, usage, name, name_codespace, description*). The relation to table SURFACE_GEOMETRY is established by the foreign keys LODx_MULTI_SURFACE_ID, where $1 \leq x \leq 4$.

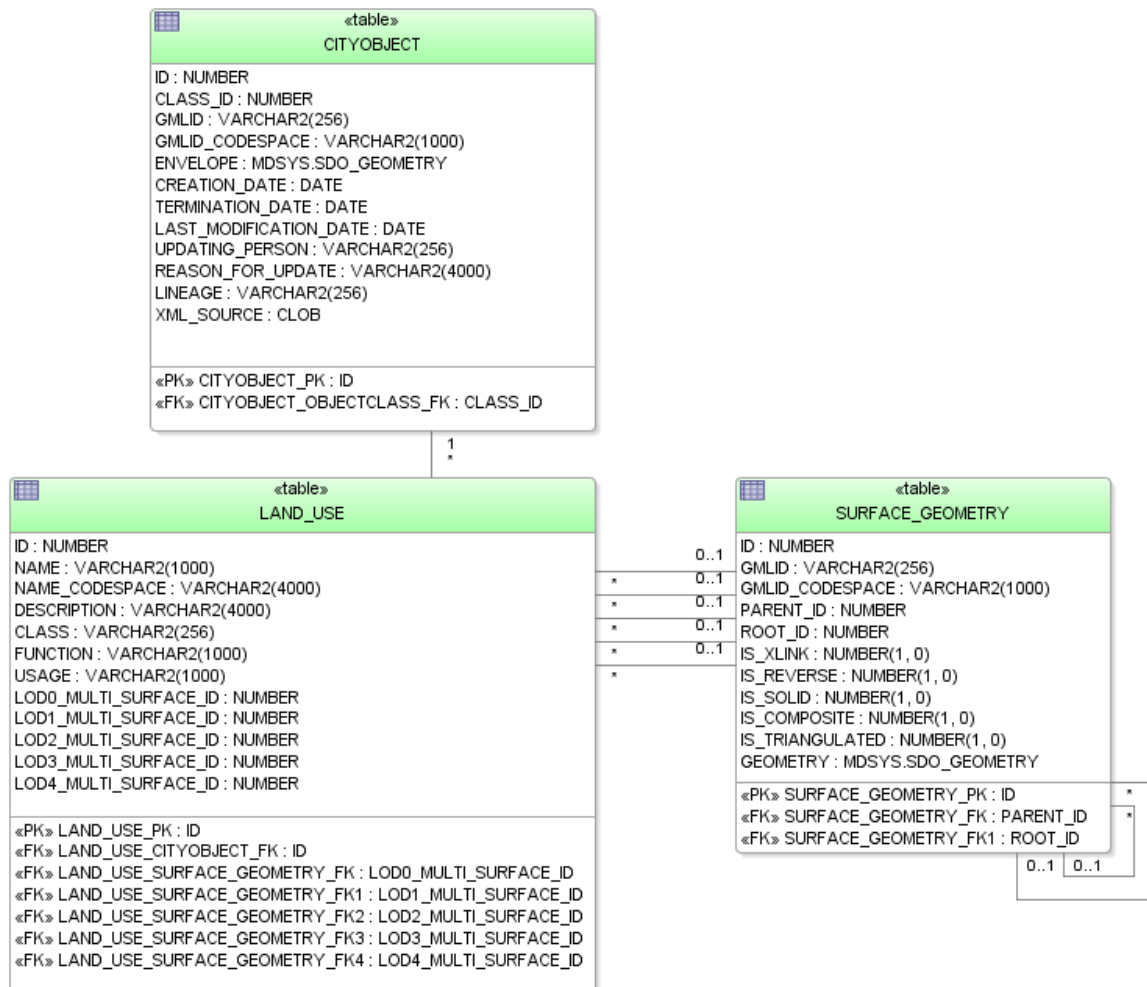


Figure 33: LandUse database schema

2.3.2.8 Orthophoto Model

The modelling of the ORTHOPHOTO database schema corresponds largely to the respective UML model. Orthophotos are not defined in the CityGML specification. For this reason orthophotos cannot be managed with the CityGML Import / Export tool. However, for managing the contents of these tables the tool for raster data import and export is available as described in chapter 7.

ORTHOPHOTO and ORTHOPHOTO_IMP

The most important attribute of table ORTHOPHOTO is ORTHOPHOTOPROPERTY, which is of type SDO_GEORASTER. It is used to store the Orthophotos that are generated by the Stored Procedure *mosaicOrthophotoInitial* from Orthophoto tiles within the import table ORTHOPHOTO_IMP. Since versioning of objects with datatype SDO_GEORASTER is unsupported by Oracle 10G, tables ORTHOPHOTO and ORTHOPHOTO_IMP remain unversioned. An Orthophoto is a CityObject and receives a foreign key to its ID attribute. The referential integrity must be maintained manually, because the version management of the Oracle Workspace Manager does not allow foreign keys from unversioned tables to versioned tables.

The attribute FILENAME contains the name of the file from which the tile was imported. The geometrical validity area or the extension of a tile is stored in the attribute FOOTPRINT. This table is also not version-enabled.

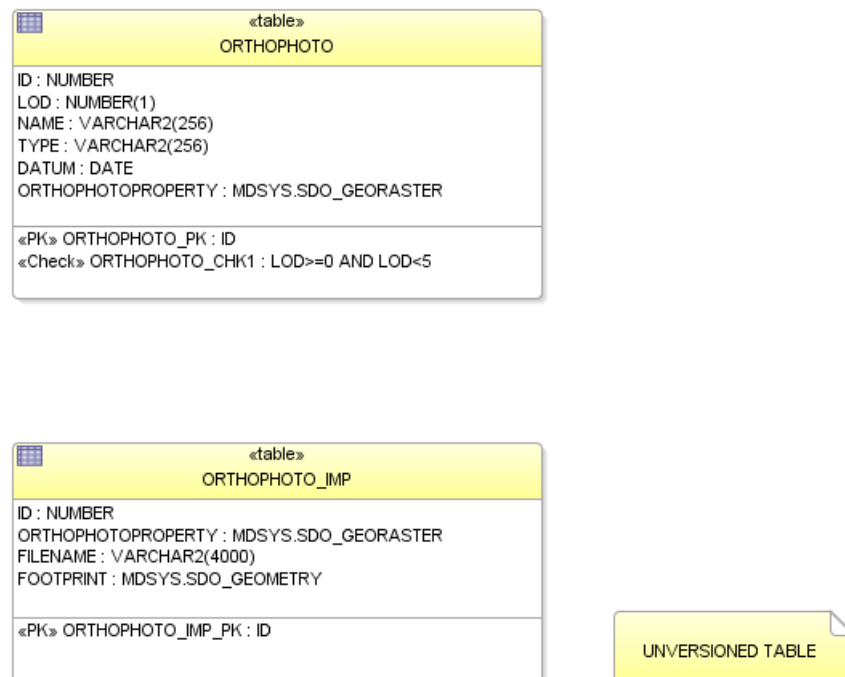


Figure 34: Orthophoto database schema

ORTHOPHOTO_RDT

The table ORTHOPHOTO _RDT is derived from the type SDO_RASTER and serves as a container for the image data of the table ORTHOPHOTO. The table is not versioned.

ORTHOPHOTO_IMP_RDT

The table ORTHOPHOTO_IMP_RDT is derived from the type SDO_RASTER and serves as a container for the image data of the table ORTHOPHOTO_IMP. The table is not versioned.

ID sequences

Following sequences are created to enable automatic increments of the IDs of objects which are not derived from CITYOBJECT:

Sequence	Table
ORTHOPHOTO_IMP_SEQ	ORTHOPHOTO_IMP
ORTHOPHOTO_RDT_SEQ	ORTHOPHOTO_RDT
ORTHOPHOTO_IMP_RDT_SEQ	ORTHOPHOTO_IMP_RDT

Table 14: Sequences for tables related to orthophotos

2.3.2.9 Transportation Model

For the realisation of transportation objects two tables are provided: TRAFFIC_AREA and TRANSPORTATION_COMPLEX.

TRAFFIC_AREA

Attributes for every traffic area object consist of *function*, *usage*, and *surfaceMaterial*. Representation of object geometry is handled by foreign keys LODx_MULTI_SURFACE_ID (with $2 \leq x \leq 4$).

Attributes of AuxilliaryTrafficAreas such as middle lanes, median strips etc. are also included in the TRAFFIC_AREA table. The attribute IS_AUXILIARY indicates whether a tuple represents a *TrafficArea* (value 0) or an *AuxiliaryTrafficArea* (value 1) feature.

TRANSPORTATION_COMPLEX

As shown in the UML diagram, every traffic area object may have the attributes *function* and *usage*. For differentiation between the subclasses *Track*, *Road*, *Railway* and *Square* the attribute TYPE is used (case sensitive!). In the coarsest level transportation complexes are modelled by line objects (attribute LOD0_NETWORK, Oracle geometry datatype MDSYS.SDO_GEOMETRY – Multicurve). Starting form LOD1 the representation of object geometry is handled by foreign keys LODx_MULTI_SURFACE_ID (with $1 \leq x \leq 4$).



Figure 35: Transportation database schema

2.3.2.10 Vegetation Model

The vegetation model specified in paragraph 2.2.1.3.8 is realized by the tables shown in figure 36 which correspond largely to the UML model.

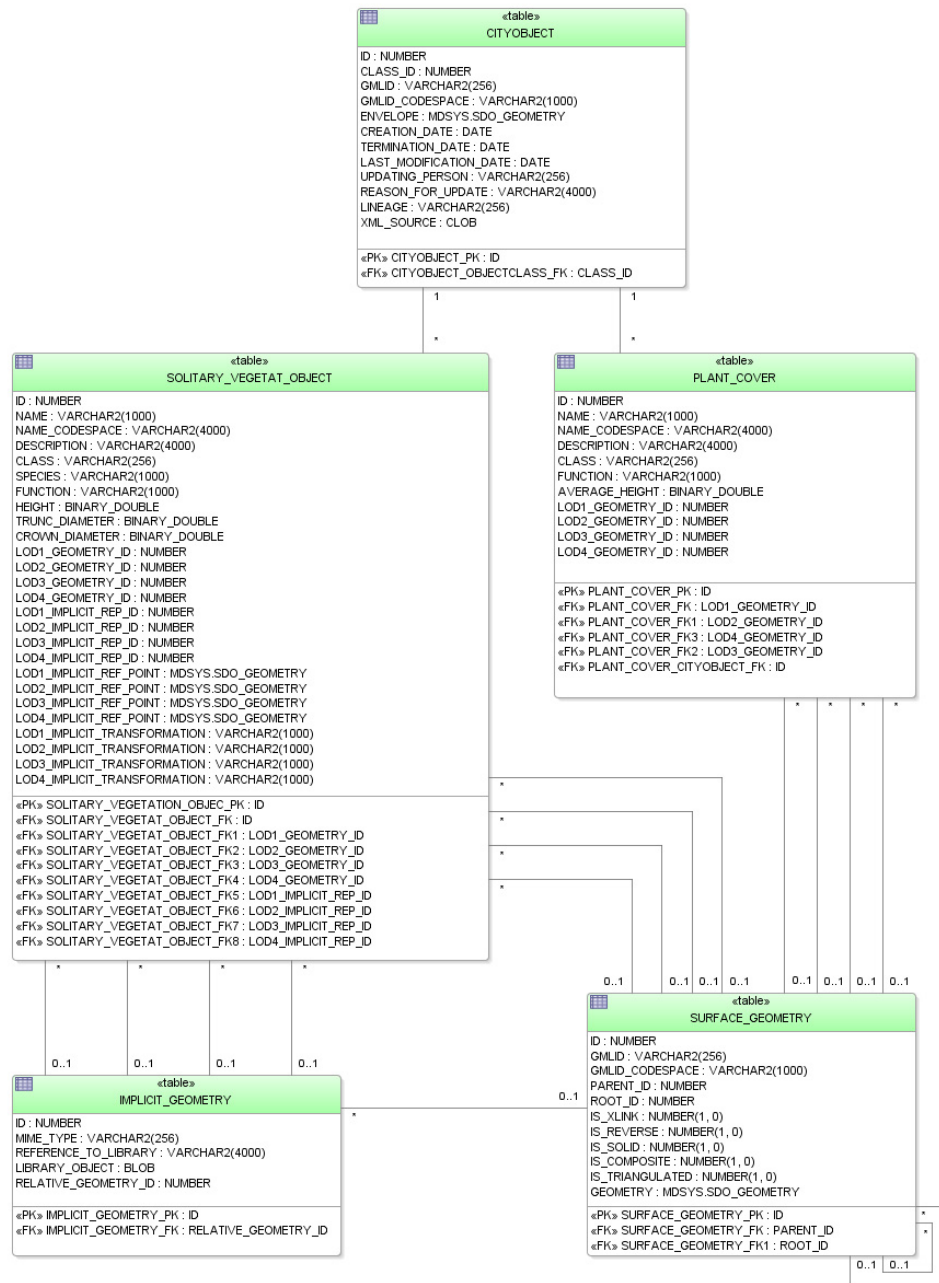


Figure 36: Vegetation database schema

SOLITARY_VEGETAT_OBJECT

The attributes *class*, *species*, *function*, *height*, *trunkDiameter*, and *crownDiameter* describe single vegetation objects.

The geometrical information can either be stored with absolute world coordinates as some surface geometry (LOD x GEOMETRY_ID, with $1 \leq x \leq 4$) or as implicit geometry. In the case of an implicit geometry, a reference point of the object (LOD x _IMPLICIT_REF_POINT, with $1 \leq x \leq 4$) and optionally a transformation matrix

(LOD_x_IMPLICIT_TRANSFORMATION, with $1 \leq x \leq 4$) must be given. In order to compute the actual location of the object, the transformation of the local coordinates into the reference system of the city model must be processed and the anchor point coordinates (LOD_x_IMPLICIT_REP_ID, with $1 \leq x \leq 4$) must be added.

PLANT_COVER

Information on vegetation areas are contained in attributes *class*, *function*, and *averageHeight*. The geometry is restricted to a MultiSurface or MultiSolid and are represented by the foreign keys LOD_xGEOMETRY_ID (with $1 \leq x \leq 4$) which refer to the SURFACE_GEOMETRY table.

2.3.2.11 WaterBody Model

WATERBODY, WATERBOD_TO_WATERBND_SRF

The modelling of the WATERBODY database schema corresponds largely to the respective UML model (*class, function, usage*). For LoD0 and LoD1 additional attributes are added, e.g. for modelling river geometry (LODx_MULTI_CURVE).

The geometries of LOD0 and LOD1 areal water bodies are stored within the table SURFACE_GEOMETRY. The foreign keys LODx_MULTI_SURFACE_ID (with $0 \leq x \leq 1$) refer to the corresponding rows in the SURFACE_GEOMETRY table

Geometry for water filled volumes is handled in a similar way using foreign keys LODx_SOLID_ID (with $1 \leq x \leq 4$).

For mapping the *boundedBy* aggregation which identifies the water body's exterior shell managed by the WATERBOUNDARY_SURFACE table the additional table WATERBOD_TO_WATERBND_SRF is needed to realise the m:n relationship.

WATERBOUNDARY_SURFACE

The *WaterBody* can be differentiated semantically by the class *WaterBoundarySurface*, which is realised by features of class *WaterSurface*, *WaterGroundSurface* or *WaterClosureSurface*. The differentiation is handled by the TYPE attribute (is case sensitive!). Since every WaterBoundarySurface object must have at least one associated surface geometry the foreign keys LODxSURFACE_ID (with $2 \leq x \leq 4$) are used to realise these relations.



Figure 37: WaterBody database schema

2.3.2.12 Sequences, Database_SRS

The table DATABASE_SRS is used to define the spatial reference system for a certain CityModel. It only contains one tuple naming the Oracle SRID and the corresponding CRS name. The information is mandatory and is entered during the database setup. SRID describes the spatial reference identifier, and GML_SRS_NAME the appropriate GML identifier (used in attribute *srsName*).

Figure 38 also lists additional data objects for sequences, which are used to generate sequential numbers to define unique primary keys automatically, and to coordinate keys across multiple rows or tables. For the database ascending sequences that start with 1 and increase by 1 with no upper limit are created.

It is highly recommended to generate ID values for all tables by using sequences exclusively.

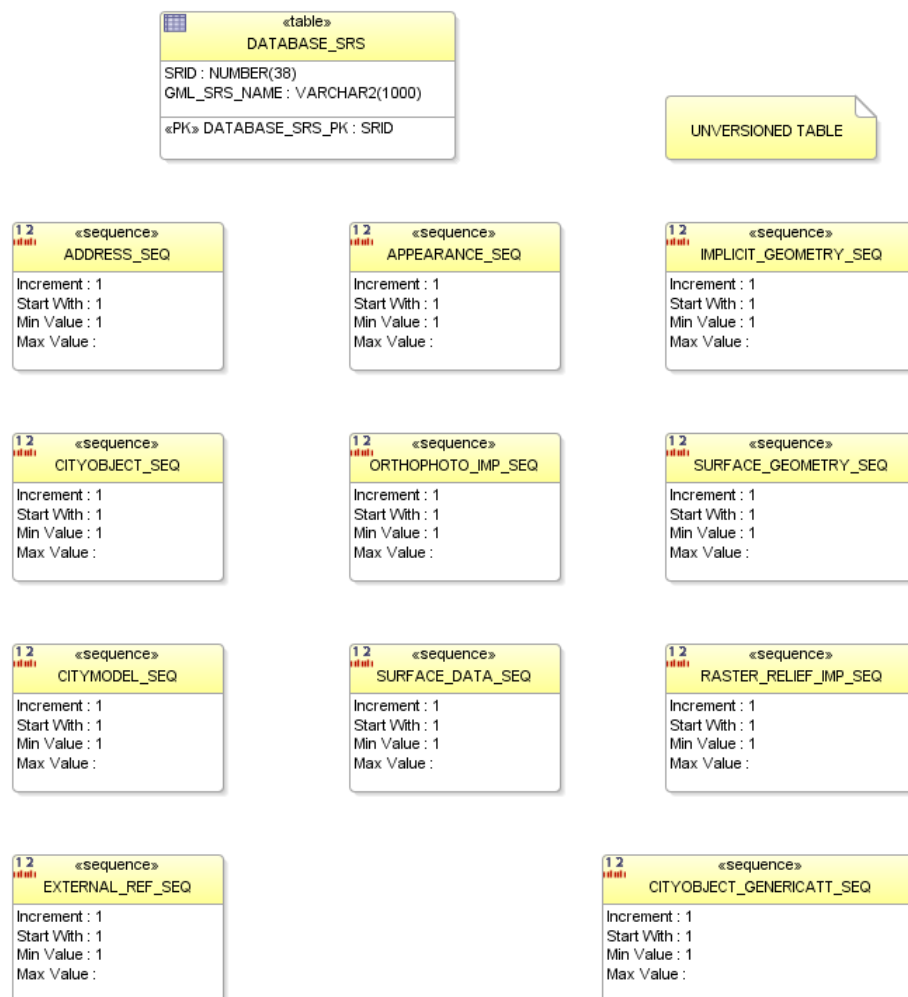


Figure 38: Tables for spatial information and sequences

3 Implementation and Installation

The utilities for implementing the 3D city database include SQL scripts for geo database setup and the CityGML import / export tool, which have been developed by the team at the Technical University Berlin. Moreover, previous developments of the workgroup at the University of Bonn - an Oracle Workspace Manager based versioning and history management and tools for import and export of raster data - are still working with version 2 of the 3D geodatabase. All files are available as zip-archives for download. They are explained in the following.

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

3.1 System requirements

Implementation of the database and the import/export tool requires access to an Oracle 10G R2 server installation. All available patches for the RDBMS should be installed to avoid unexpected errors. In fact, they **must** be installed if version management or the matching tool will be used. The tools are OS independent.

Multiprocessor systems or multicore CPUs will help to fully exploit the potential of the database and especially of the import/export software. Of course a high amount of main memory will speed up the tools leading to high performance processing.

The CityGML import/export tool requires Java Runtime Environment Standard Edition (SE) version 1.6.0 Update 5 (JRE 1.6.0) or higher. JRE 1.6.0 can be downloaded via the following link

<http://java.sun.com/javase/downloads/>

and must be installed prior to the execution of the import/export tool. Please make sure that you download the correct JRE according to your operating system (e.g. Windows, Linux, Solaris, Apple OS X). JRE 1.6.0 requires approximately 15 MB of hard disk space.

The CityGML import/export tool needs approx. 13 MB of hard disk space and requires at least 256 MB of main memory (RAM). This is sufficient for importing and exporting of CityGML files (recommended size: smaller than 10 MB) or raster data (recommended size: smaller than 50 MB). For the import of large CityGML files or big images, a minimum of 1 GB of main memory is recommended.

Please note that additional hard disk space is required for the storage of exported CityGML files. After completion of the export process, these files can be relocated. Furthermore temporary hard disk space is required for the tiling of raster DEMs or orthophotos, depending on the size of the image to be tiled.

3.2 Database setup

3.2.1 SQL scripts

Database generation is managed by a variety of scripts, which are stored within the distribution package. To setup the database and to control the execution of system operations several privileges have to be assigned to a user. Managing and controlling privileges is made easier by using roles introduced in Oracle 10G. Following predefined roles are necessary: CONNECT and RESOURCE. As Oracle has removed some privileges from the CONNECT role in version 10.2, at least CREATE TABLE and CREATE SEQUENCE privileges have to be explicitly granted to the DB user account in addition.

Calling `CREATE_DB` from within the top-level SQL directory will start the setup procedure. At first, the user has to enter two mandatory parameters that define the selected spatial reference system: Oracle's spatial reference identifier (SRID) and the appropriate GML identifier (srsName) as specified and required by the OGC. This may refer to the database of predefined CRS provided by the European Petroleum Survey Group (EPSG). You will find detailed information in the Best Practices Paper *Definition identifier URNs in OGC namespace* of the OGC [Whiteside 2007]. The default values should only be accepted if the user intends to setup the database for Berlin with the Oracle SRS 81989002 (which is a user-defined CRS) and the String 'urn:ogc:def:crs:crs:EPSG:6.12:3068:EPSG:6.12:3068:5783'. Please execute the SQL-script `SOLDNER_BERLIN_SRS_10G_R2.sql` in folder `SYSDBA` if the Soldner system is undefined. The Universal Resource Name (URN) also defines Berlin's reference system and is composed of the Strings EPSG:3068 (DHDN / Soldner Berlin - planimetry / horizontal reference system), the version of the EPSG database (6.12), and EPSG:5783 (DHHN92 vertical reference system). For further details on reference systems and codes which are appropriate to your application see <http://www.epsg-registry.org/> or <http://spatialreference.org/>.

Subsequently the user can decide whether to enable versioning. However, this step can also be caught up later, using the SQL script `ENABLEVERSIONING.sql`. Calling `DISABLEVERSIONING.sql` disables versioning, which is highly recommended if the database will initially be filled with a huge number of objects.

Further steps are carried out automatically. Besides the creation of the tables indexes are established which are necessary for performance enhancements. This concerns indexes for primary and foreign keys and spatial indexes for geoobjects. During the installation process, tables and procedures for the PlanningManager are setup automatically. Calling the script `DROP_PLANNINGMANAGER` removes the PlanningManager, if it is not needed.

Moreover, SQL scripts are available for calling the mosaic function of Oracle GeoRaster, for gathering orthophoto tiles within one large raster data object for a given LoD (MOSAIC), and for creating a report on the objects stored in the database (GEODB_REPORT).

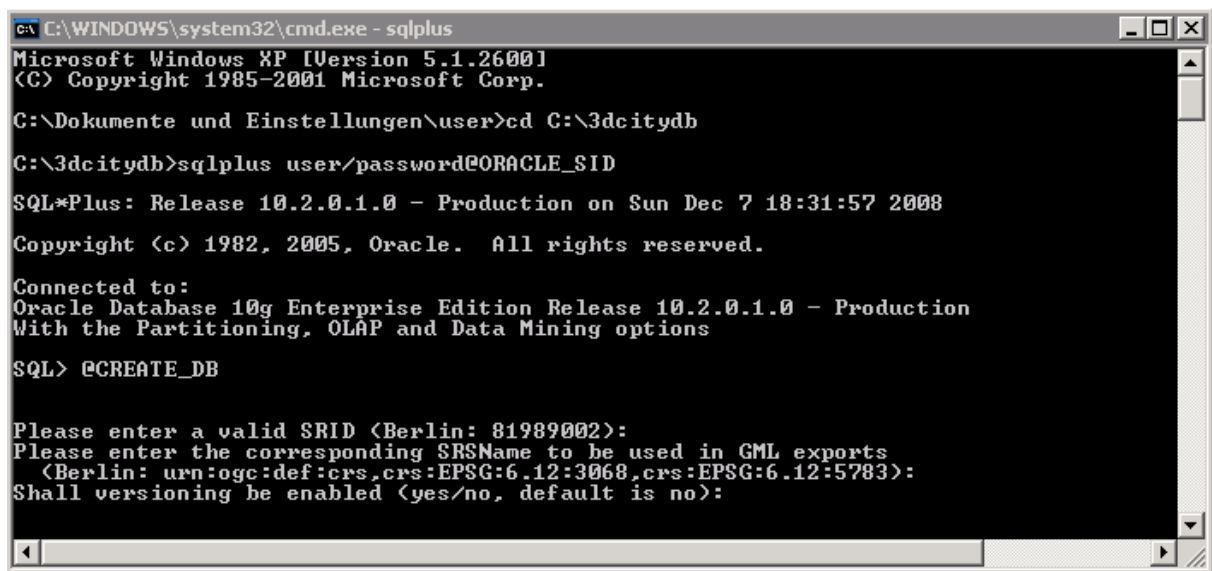
If the user intends to delete the entire database (with help of `DROP_DB`) the following sequence is recommended:

@DISABLEVERSIONING.sql – for shutdown of versioning
@DROP_DB.sql

For your information all scripts are listed in Appendix A.

3.2.2 Example session

To install the database schema, first unpack the installation package to some directory. Start the command shell and change the working directory to the directory containing the unpacked SQL scripts. Connect to the ORACLE DBMS with sqlplus. Start the script CREATE_DB.sql as shown in figure 39. A more detailed output is listed in Appendix B.



```
C:\WINDOWS\system32\cmd.exe - sqlplus
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Dokumente und Einstellungen\user>cd C:\3dcitydb
C:\3dcitydb>sqlplus user/password@ORACLE_SID

SQL*Plus: Release 10.2.0.1.0 - Production on Sun Dec 7 18:31:57 2008
Copyright (c) 1982, 2005, Oracle. All rights reserved.

Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.1.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> @CREATE_DB

Please enter a valid SRID (Berlin: 81989002):
Please enter the corresponding SRSName to be used in GML exports
(Berlin: urn:ogc:def:crs,crs:EPSG:6.12:3068,crs:EPSG:6.12:5783):
Shall versioning be enabled (yes/no, default is no):
```

Figure 39: Example for database setup

4 CityGML Import / Export Tool

The 3D City Database Import/Export Tool is a Java based front-end for the 3D City Database version 2.0. It allows for high-performance importing and exporting spatial data for a virtual 3D city model.

Its main characteristics are:

- Full support for CityGML version 1.0.0, 0.4.0, 0.3.1, and 0.3.0 (the latter two only for reading)
- Reading/Writing CityGML instance documents of arbitrary file size
- Multithreaded programming facilitating high-performance CityGML processing
- Support of the CityGML appearance model
- Resolving of XLinks
- XML validation of CityGML instance documents

The 3D City Database Import/Export Tool comes with both a Graphical User Interface (GUI) for end-user interaction and a Command Line Interface (CLI). The latter one allows for employing the tool in batch processing workflows or embedding its functionality into third party programs.

As mentioned before the import / export tool requires

- Java JRE or JDK version 1.6.0_05 or higher
- Oracle Spatial DBMS version 10G R2 or higher
- 3D City Database version 2.0

The 3D City Database Import/Export Tool is free software under the GNU Lesser General Public License Version 3.0. See the file LICENSE shipped together with the program for more details. For a copy of the GNU Lesser General Public License see the files COPYING and COPYING.LESSER or visit <http://www.gnu.org/licenses/>.

The application files can be downloaded from the website of the Institute of Geodesy and Geoinformation Science (IGG) at Technische Universität Berlin, Germany. You will find a universal installer, sources and documents here: <http://www.igg.tu-berlin.de/software/>. The universal installer can be used for different operating systems and has been tested on Windows XP SP3, Windows Vista, and MacOS X.

It is recommended to use the universal installer to unpack the 3D City Database Import/Export Tool application files to your local computer. Afterwards, you can immediately run the application. Once unpacked, you can always copy & paste the application files to any other location without using the universal installer. However, please make sure that at least the following subfolders and their contents are in the same folder as the application .jar file (impexp.jar):

```
--(parent folder)
  |--lib
  |--schemas
  |--config
  |--log
  impexp.jar
```

There is no uninstallation routine needed to remove the 3D City Database Import/Export Tool. Simply delete all application files from your local computer.

4.1 Import / Export Tool Interfaces

For running the 3D Database Import/Export tool use one of the following options:

- a) Recommended: Use of graphical user interface (GUI)
- b) Command line interface (CLI)

4.1.1 Command line interface

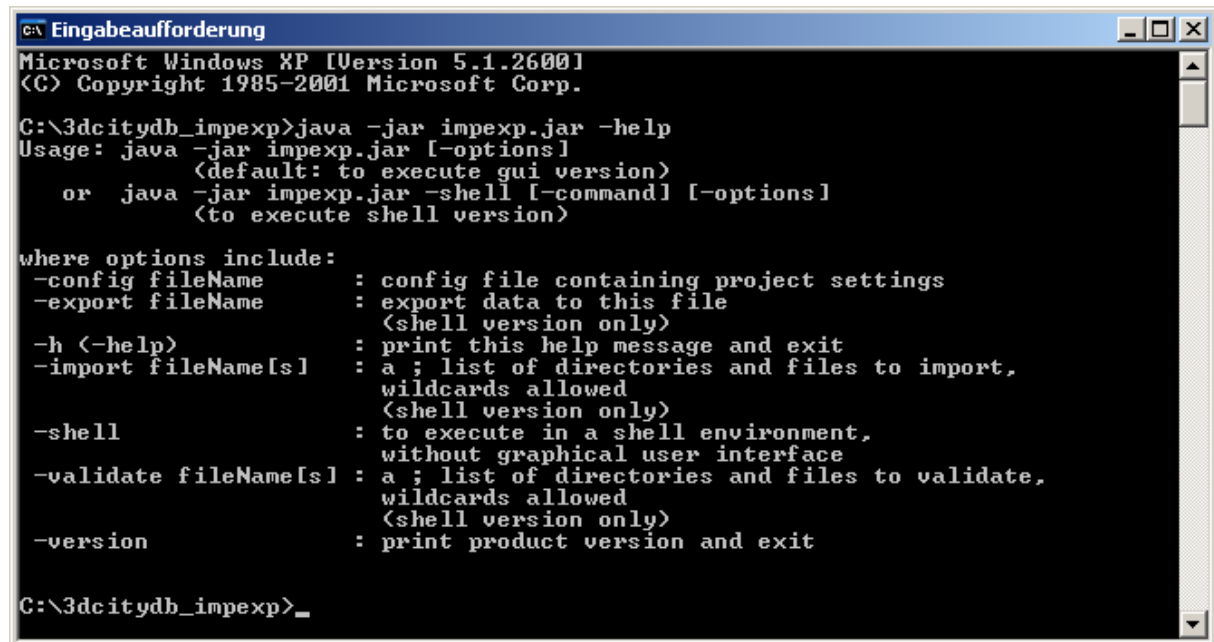
Open a Shell and type the following:

```
java -jar impexp.jar [-options]
```

Again, define the memory limits for the JVM. See the starter scripts stored in the installation directory (start_unix.sh or start_win.bat) for examples. Take note of the default values for the -Xms and -Xmx option, specifying the initial and maximum size of the memory allocation pool. Type

```
java -jar impexp.jar -help
```

to get a list of supported command line parameters:



```

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\3dcitydb_impexp>java -jar impexp.jar -help
Usage: java -jar impexp.jar [-options]
        <default: to execute gui version>
    or  java -jar impexp.jar -shell [-command] [-options]
        <to execute shell version>

where options include:
  -config fileName      : config file containing project settings
  -export fileName      : export data to this file
                        <shell version only>
  -h <-help>           : print this help message and exit
  -import fileName[s]  : a ; list of directories and files to import,
                        wildcards allowed
                        <shell version only>
  -shell               : to execute in a shell environment,
                        without graphical user interface
  -validate fileName[s]: a ; list of directories and files to validate,
                        wildcards allowed
                        <shell version only>
  -version             : print product version and exit

C:\3dcitydb_impexp>_

```

Figure 40: CLI help text

Program initialisation is by default controlled by the configuration file (project.xml) stored in the “config” subfolder of the installation directory. In this file the information concerning the database connection has to be included (server name, database name, port, username and password). The structure of the configuration file used in CLI mode follows the structure used in GUI mode. Hence, it is recommended to define the settings using the GUI and to store the config file as describe in the next chapter.

You can also override this default behaviour and provide a different configuration file using the -config option (this option is available both for the GUI and the CLI version). The project settings contained in this file must also follow the general XML structure for configuration files defined by the Import/Export tool.

4.1.2 Graphical User interface

Alternatively to the CLI the graphical Import / Export user interface can be used. It is implemented in German and English language and is explained in the following.

In order to start the 3D Database Import/Export Tool GUI please check the following steps:

- Edit the starter scripts according to your needs (in particular, **adapt the memory limits**). Two starter scripts are included in the installation folder:
 - start_unix.sh (use for UNIX/Linux and derivates, MacOS X, etc.)
 - start_win.bat (use for MS Windows family, tested for XP and Vista)
- Double click onto the starter script.

You also can double click onto the impexp.jar file only. **However, this is not the recommended way of starting the application since you may quickly run into heap space memory lacks due to low JVM default values.**

After program startup the main menu bar contains the **File | Project | Help** commands in the main menu bar. The **Project** menu allows for loading project settings from an external XML-based configuration file. All current settings will be overridden with the settings contained in this file. However, the external file will be rejected if it is not a valid XML document with respect to the XML schema definition for configuration files defined by the Import / Export tool. Furthermore, the **Project** menu allows for saving the current project settings to a configuration file.

If you want to create your own configuration file, for example for usage with the CLI version of the Import / Export tool, it is recommended to use the GUI version to choose appropriate settings and afterwards to export the settings to an external file. By this means, you can ensure that the exported configuration file is valid. However, you can also create a configuration file manually. The **Project** menu offers the possibility to store the XML schema document for configuration files to your local computer. Please make sure to validate your configuration file against this schema prior to using it with the Import / Export tool.

The **Help** menu allows for opening a dialog containing general information about the Import / Export tool. Amongst others, this includes the official version string of the program.

4.1.2.1 Database connection

For direct access to the Oracle database authentication and database information is mandatory. This information is collected in the **Database** form. First, mandatory for authentication, the user must provide a valid username and password. If he intends to store the password activation of the checkbox is necessary. Please note, that the password will be stored as plain text in the configuration file. Server specifications such as server name and the port number of Oracle's default listener (default: 1521) are also mandatory. As there can be more than one Oracle instance on a server machine a System Identifier (SID) or SERVICE_NAME depending on the database installation is necessary in order to be able to distinguish these instances. If the user intends to set up a new database the connection name, a brief description and workspace name have to be added.

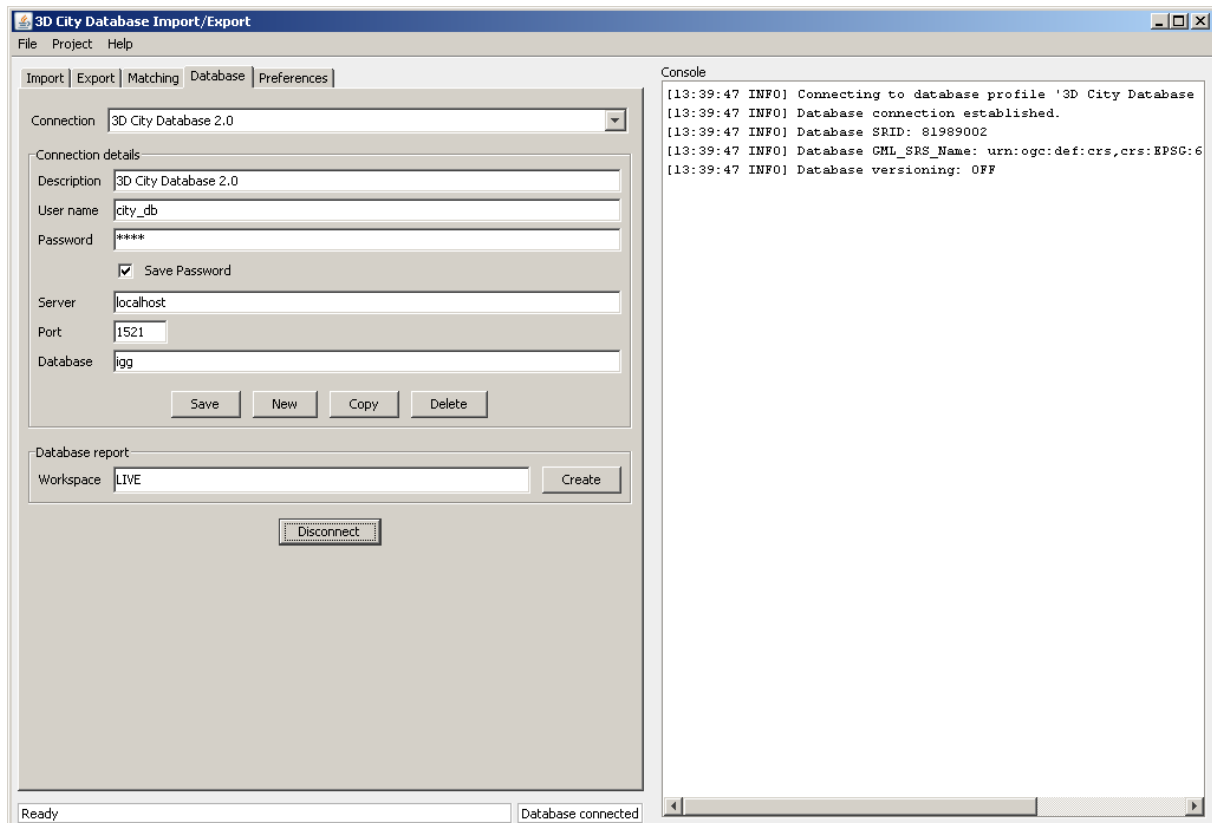


Figure 41: Database connection

The **Connect** button connects / disconnects the tool with / from the database.

The tool allows management and administration of several accounts which you can create (**New**), copy or delete using the appropriate button.

Messages, warnings and errors are logged to the console window. Status information is displayed in the status bar at the bottom of the window.

4.1.2.2 CityGML File Import

The import functionality is used for adding new records to the 3D geodatabase. CityGML objects are added to the database as new objects.

Special attention is directed to the management of object IDs, because only a systematic and careful usage of a worldwide unique identifier guarantees consistent updating of geo objects stored in the database. However, GMLIDs are optional and unique only within one dataset. To overcome this problem, two alternative strategies are pursued during import of objects to the database:

- all objects are assigned a newly generated worldwide unique identifier (UUIDs) or
- only objects with missing GMLIDs are assigned such UUIDs.

Settings defined in import preferences control which procedure is applied (cf. Figure 45). In either case, the attribute GMLID_CODESPACE described in section 2.3.2.1 is added to every feature.

During the import procedure existing records in the database are not replaced. Hence the import tool will be not sufficient for updating stored objects.

Click on the **Import** tab to start the import of CityGML objects. You will find a window subdivided into three parts for selecting directories and files, defining filter operations, and for monitoring the program flow.

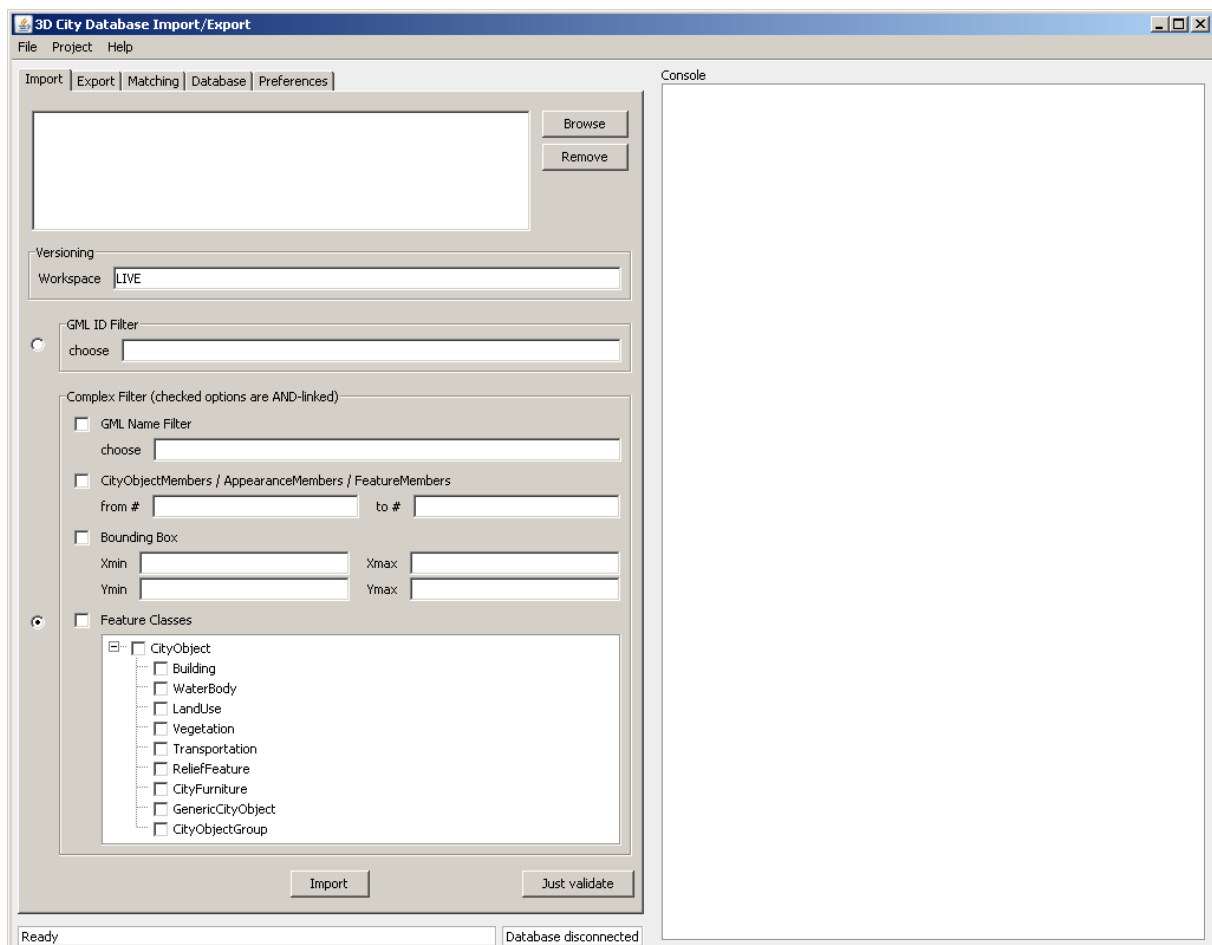


Figure 42: Graphical User Interface for data import

File selection

File selection can be handled in one of two ways, by

- selecting the file in the operating system's file manager window. Drag and drop the designated file(s) into the import panel. You can either replace all previous files in the list with those dragged into the import panel (default) or you can append them (on Windows systems keep CTRL pressed while dropping).
- pushing the **Browse** button, which opens the Folder List provided by the operating system. Select the file(s) of interest as usual.

More than one file may be selected. All files will be imported. Files can be removed from the list by selecting the corresponding file name(s) in the list and clicking on the **Remove** button.

Versioning

Oracle's Workspace Manager enables storing of different versions of the database as named workspaces. The workspace to which the CityGML file should be imported can be explicitly named. Default is 'LIVE' (which must also be used, if version management is disabled).

Import Filter

Radio buttons are used for selecting objects to be included in the database. Two main alternatives allow control of data import:

- **Import** *GML-ID filter*
Enter the GMLIDs of the object(s) of interest. Multiple IDs are separated by commas. Only these objects will be imported from the file(s).
- **Import** *Complex filter*
This feature controls data import and permits multiple selection criteria from a number of filter type options. Any combination, handled as logical AND operation, is determined by activating the corresponding checkboxes. If none of them is selected filtering is not applied.
 - *GML Name Filter*
By selecting a gml:name the object of interest is requested from the CityGML file and imported including all subfeatures, provided that the search succeeded. ATTENTION: Only a single GML name is accepted!
 - *CityObjectMembers / AppearanceMembers / FeatureMembers*
If you plan to import only a subset of objects, appearance members, or features, you can restrict the import to n top level features starting at feature m followed by the next n elements in the dataset. For example, if only the first 1000 features should be imported, the values $m=1$ (from #) and $n=1000$ (to #) have to be inserted.
 - *BoundingBox*
For importing objects in an area of interest (AoI), lower left and upper right coordinates of the bounding box are needed. According to the predefined bounding box options (cf. figure 48), objects are imported if they partly or fully overlap the AoI.
 - *FeatureClasses*
Relevant feature class(es) are imported by selecting the corresponding checkboxes. Only top-level feature classes can be selected. If e.g. buildings are chosen, then also the contained features like BuildingParts, WallSurfaces etc. are imported.

After selection of filter parameters, simply press the **Import** button to start the import procedure.

XML Validation

The Import / Export tool allows for validating the input CityGML instance documents against the official CityGML XML schema documents. By this means, you can ensure that the files to be imported are valid with respect to the CityGML schemas. Invalid files might cause the import procedure to behave unexpectedly or even to abort abnormally. Thus, it is **strongly recommended** to only import valid CityGML instance documents as this improves the robustness of the Import / Export tool.

The XML validation process is started by pressing the **Just validate** button. As the name of this button already indicates, the selected files will just be validated and therefore will **not** be imported into the database. The results of the validation process are printed to the console window.

The behaviour of the XML validation process can be customized through according settings in the preferences tab (cf. Figure 49).

4.1.2.3 CityGML Export

The Export function generates a CityGML formatted output data file for all CityGML object types which are modeled in the Oracle database. Similar to the import the user is able to export only a selected set of objects. Like for data import, selection filters allow the extraction of single objects (ID), objects within a certain area defined by a bounding box, or of specific feature classes.

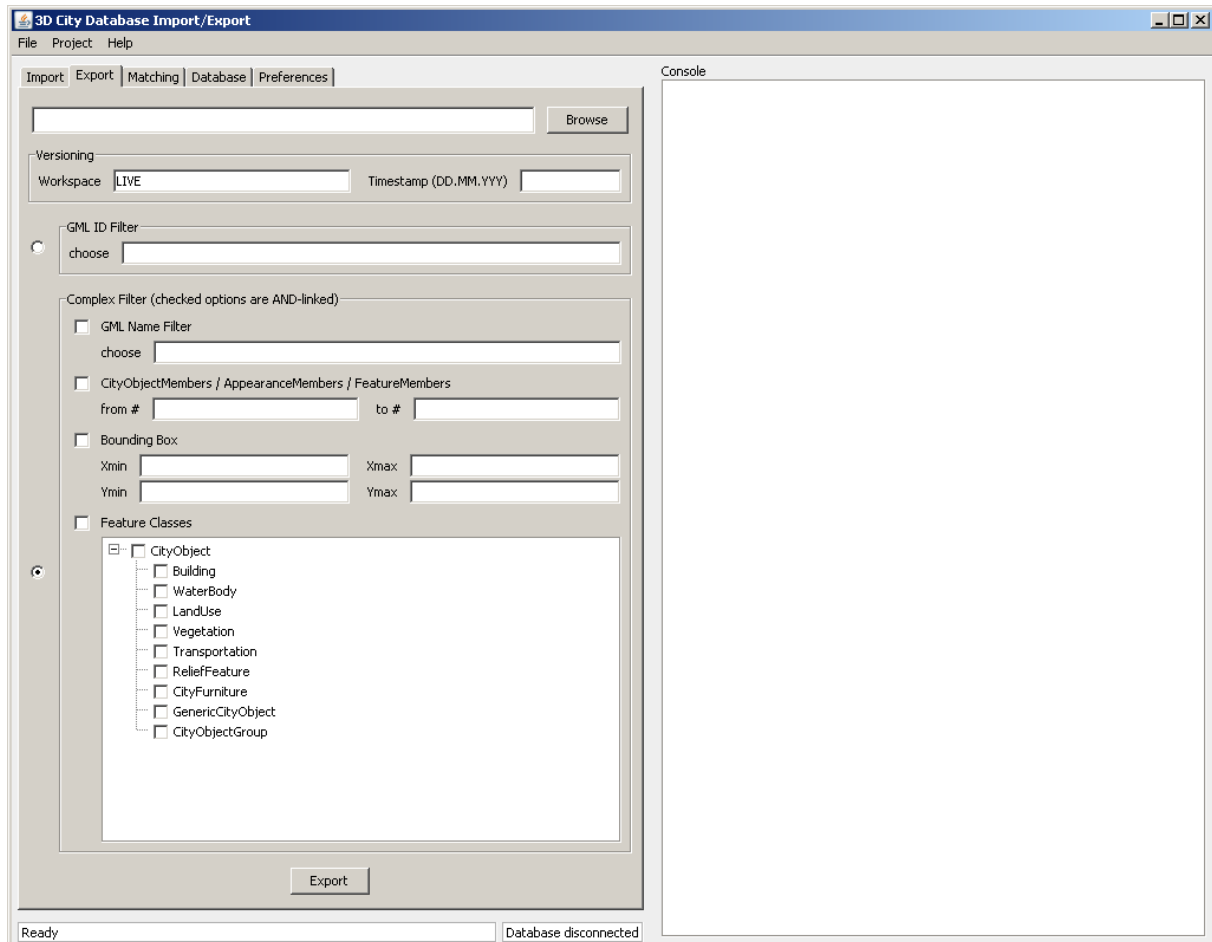


Figure 43: Export dialog window

Output file selection is similar to data import.

- Type the filename directly into export panel or
- Activate the file dialog provided by the operating system after pushing the Browse button. Select the output folder, enter a new filename and confirm the request.

Versioning

The Workspace Manager provided by Oracle is a comprehensive tool for version and history management. If the workspace management is activated, it works widely transparent for applications connected to the database. Enter a timestamp for saving the database at a specific date for documentation or for later reproduction of any processing state available. If version management is disabled or the current state of the database should be exported, the default workspace name 'LIVE' must be entered and the timestamp field must remain empty.

Output filter

Filtering works similar to the import procedure. Again the choice between two alternatives is available:

- **Export** *GML-ID filter*
Enter the GML IDs of the object(s) of interest. Multiple IDs are separated by commas. If the GML ID refers to an object group, all group members are exported.
- **Export** *Complex filter*
Any combination of the four different filter types controls the selection. The favoured combination is determined by activating the corresponding checkboxes. If none of the checkboxes is activated none of the filter operation is applied!
 - *GML Name Filter*
Enter the GML name of the object of interest. It is requested from the database and exported including all aggregated subfeatures if the query succeeded. Only a single GML name is accepted. Wildcards are supported indirectly. In the program code GML names are tested by the SQL clause “GMLNAME LIKE ‘%name%’”, which means that in case of ‘castle’ as input value, all features whose *gml:name* contains the word ‘castle’ are selected.
 - *CityObjectMembers / AppearanceMembers / FeatureMembers*
Enter a subset of objects identified by start ID=*m* followed by the next *n* top level features as defined in table CITYOBJECT.
 - *BoundingBox*
Enter the coordinates of a bounding box defining the area of interest. According to the predefined bounding box options (cf. figure 52), objects are exported if they are partly or fully covered by the specified bounding box.
 - *FeatureClasses*
Feature class(es) of relevance are exported by selecting the corresponding checkboxes. Only top-level feature classes can be selected. If e.g. buildings are chosen, then also the contained features like BuildingParts, WallSurfaces etc. are exported. If only CityObjectGroup is activated, all groups and group members are selected. If CityObjectGroup and in addition features are selected, all groups containing these features and also all selected features (which are not group members). Example: CityObjectGroup and Building are activated. The result will contain
 1. all groups which contain buildings as member and
 2. all buildings which are not part of a group.

After selection of filter parameters, click on the **Export** button to start the export procedure.

4.1.2.4 Preferences

The dialog shown in Figure 44 opens if a user selects the **Preferences** tab. The set of parameters influences the data flow and allows customizing the Import / Export tool. Settings are saved within an XML document (`project.xml`) in the in the tool's `config` folder. These preferences are automatically loaded during program launch.

Three categories control Import, Export, and General settings.

4.1.2.4.1 Import preferences

Continuation / Metadata for Updates

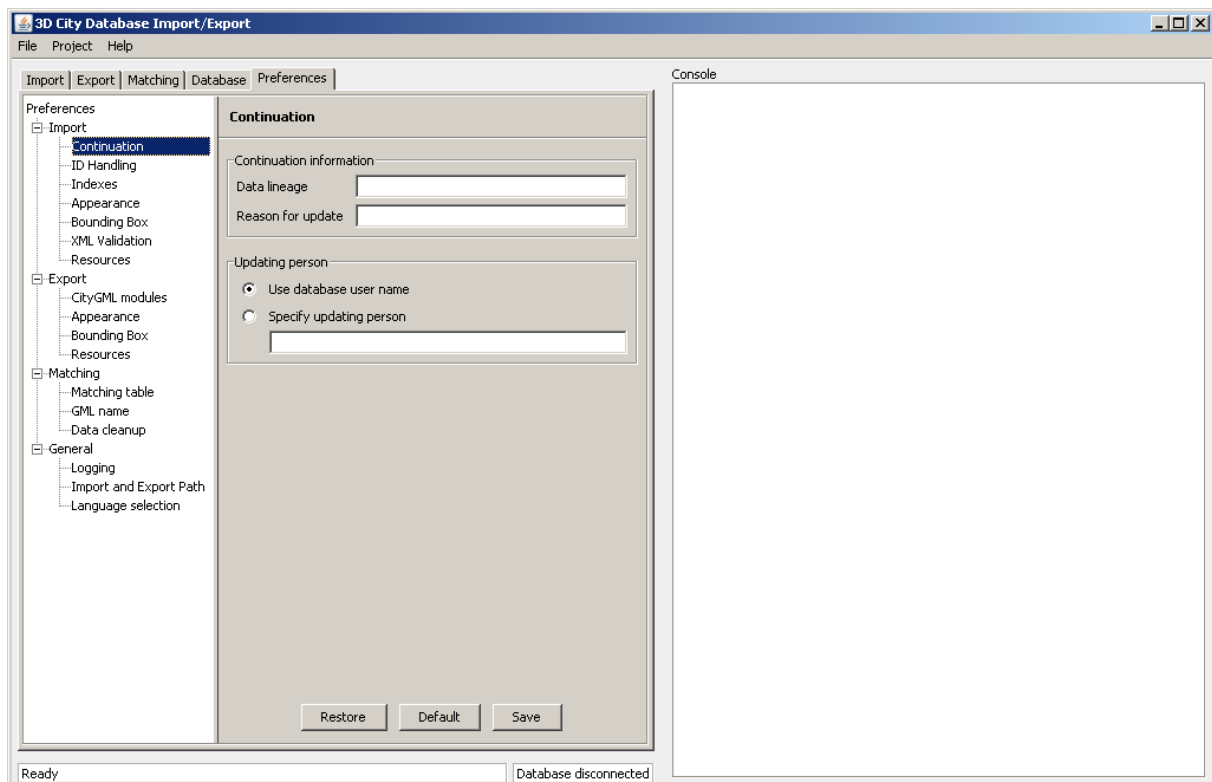


Figure 44: Import preferences – Continuation

Metadata are needed to provide additional information about the data source. Data origin and reason for update can be added as well as the information about the person responsible for executing the task. These metadata are stored for each tuple to be inserted into table CITYOBJECTS. Complete the form by specifying following information:

- Continuation information:
 - Data lineage (will be stored in attribute LINEAGE)
 - Reason for update (will be stored in attribute REASON_FOR_UPDATE)
- Updating person (will be stored in attribute UPDATING_PERSON)
 - Use database user name (default setting)
 - Specify updating person

Use buttons **Restore**, **Default** or **Save** for restoring, resetting or storing preference values.

ID Handling

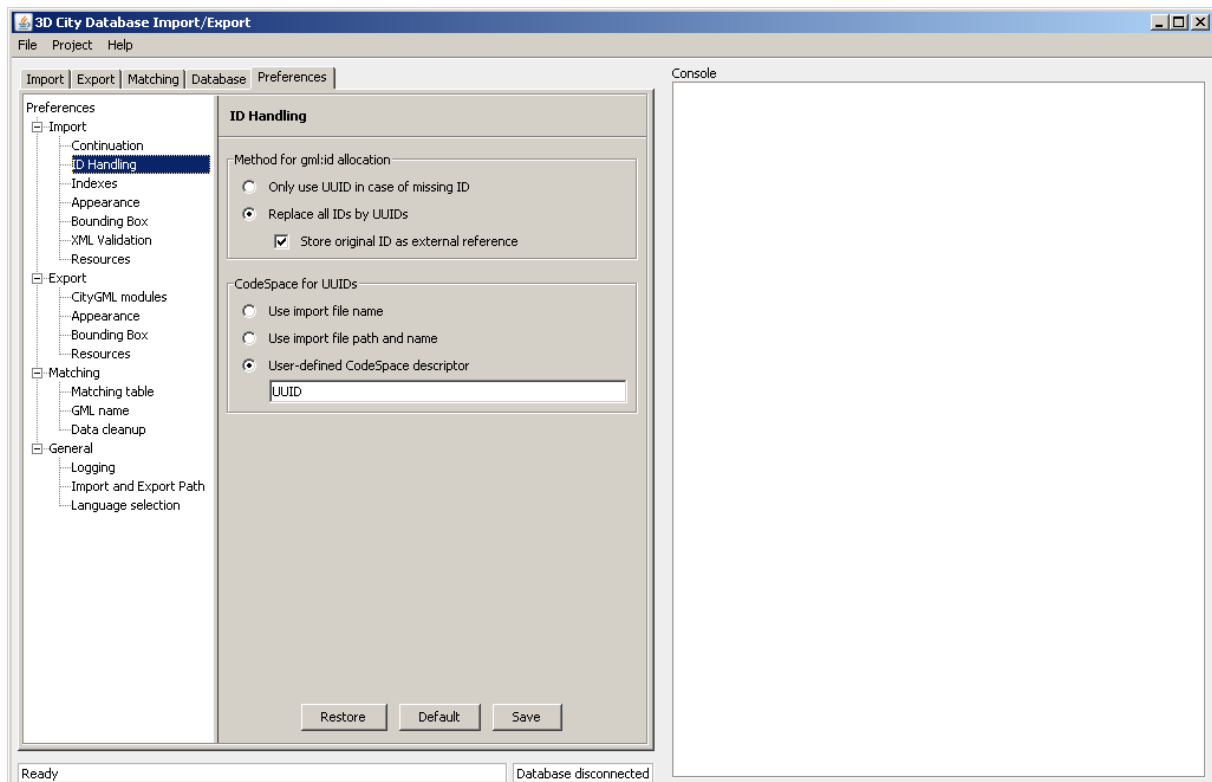


Figure 45: Import preferences – ID Handling

Unique object identifiers are essential in order to avoid name conflicts in case of future updates. Uniqueness can be guaranteed by automatically generating a ‘Universally Unique Identifier’ (UUID).

- Method for gml:id allocation:
 - Only use UUID in case of missing ID: all GML objects (features and surface geometries) that do not have a gml:id attribute will receive a newly generated UUID value.
 - Replace all IDs by UUIDs (default setting)
 - Store original ID as external reference (only for CityObjects)
- CodeSpace for UUIDs
 - Use import file name
 - Use import file path and name
 - User-defined CodeSpace descriptor (default setting)
 - Proposed value: UUID

Please note, that it is highly recommended to replace IDs with UUIDs if the dataset to be imported does not contain globally unique gml:id values so far.

Use **Restore**, **Default** or **Save** for restoring, resetting or storing preference values.

Indexes

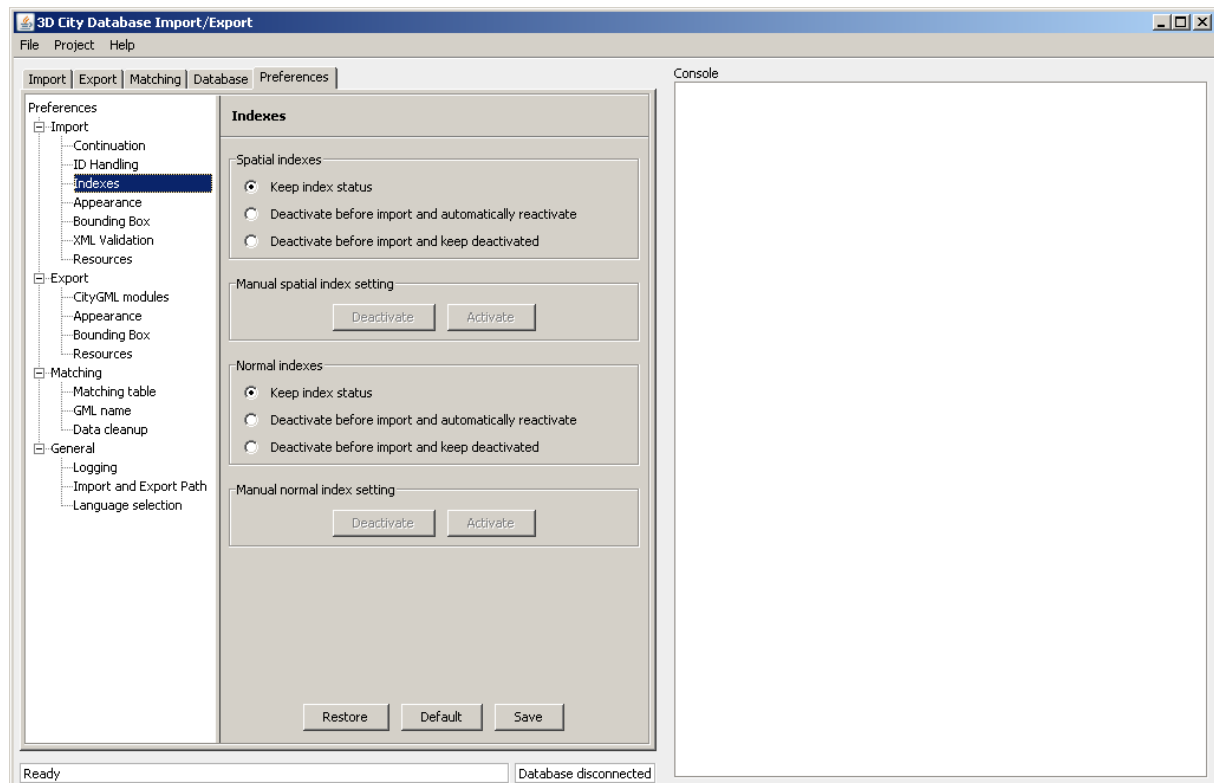


Figure 46: Import preferences – Spatial and normal Indexes

Indexes speed up access to data and therefore shorten execution time of a query because full table scans can be avoided. However, index structures have disadvantages with respect to database updates, because they must be updated each time the data is modified. Thus, the import of a large amount of objects could be highly protracted.

Based on the fact that the set up of a new database is performed infrequently, indexing is activated automatically during the initial installation process.

Preference settings in this window allow activation and deactivation of indexes. For increasing efficiency the following recommendations are given:

- Activate indexing or keep indexing active, if few objects are to be imported only
- Deactivate indexing before importing a big amount of data, reactivate indexing after the procedure is complete (and enjoy a cup of coffee – or even a lunch, depending on the amount of data).

Normal and spatial indexes are treated separately.

The choice list is more or less self-explaining. Use the radio buttons in order to select the favoured operations and click the relevant buttons for activation. But keep in mind: recreation of indexes can take several hours for large databases!

- Spatial index handling while database import
 - Keep index status (default setting)
 - Deactivate before import and automatically reactivate
 - Deactivate before import and keep deactivated
- Manual spatial index setting (only available if the connection to the database is established)
 - Deactivate
 - Activate
- Normal index handling while database import
 - Keep index status (default setting)
 - Deactivate before import and automatically reactivate
 - Deactivate before import and keep deactivated (in this case, manual reactivation is necessary; use this option when importing several files consecutively)
- Manual normal index setting (can only be activated, if the database is connected)
 - Deactivate
 - Activate

Please note that activating and deactivating of indexes cannot be aborted by the user as this might lead to an inconsistent database state.

Use **Restore**, **Default** or **Save** for restoring, resetting or storing preference values.

Appearance

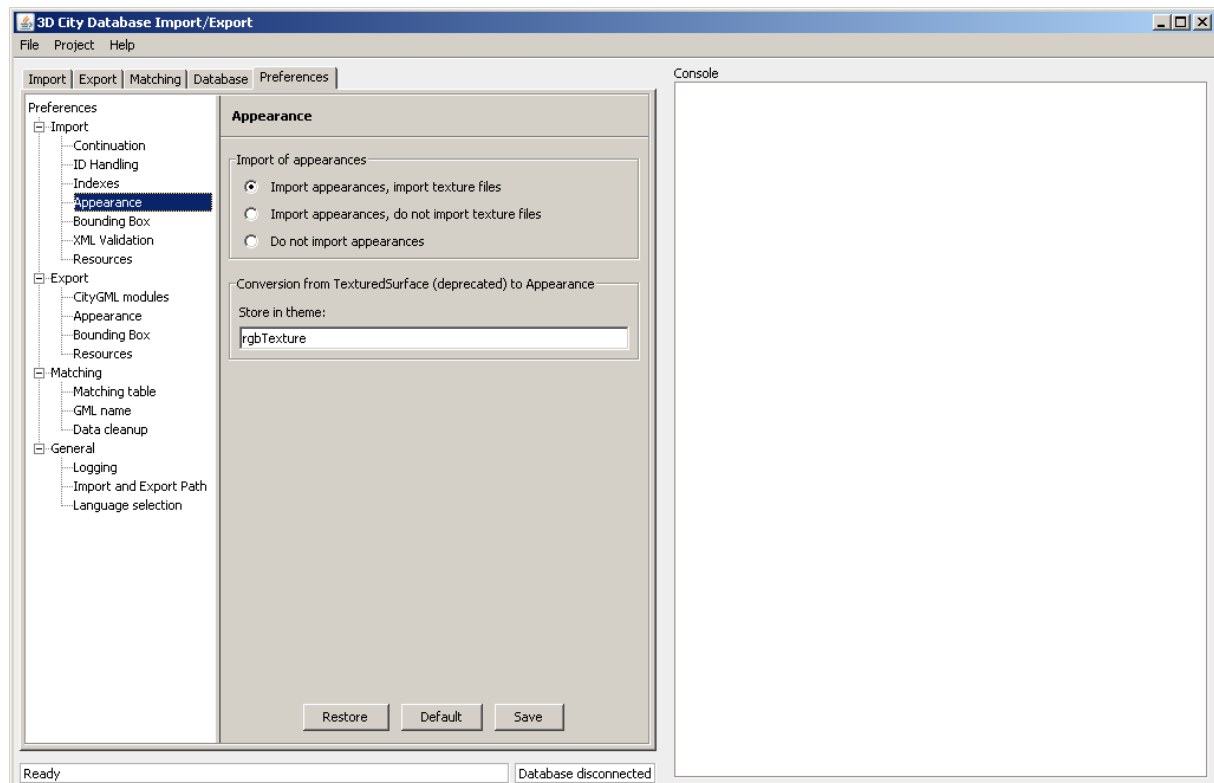


Figure 47: Import preferences – Appearance

In addition to geometry, topology, and semantics, CityGML features can be assigned appearances. Import and storage of appearances and dedicated textures are controlled by activating the appropriate radio button.

It is generally recommended not to use TexturedSurface elements anymore, because they are deprecated since CityGML version 0.4.0. Apply the concepts of the CityGML Appearance module instead. However, TexturedSurfaces can be imported. They are automatically transformed to the new appearance model.

Each surface may be assigned one or more appearances, each referring to one side of the surface.

- Import of appearances
 - Import appearances, import texture files (default setting) – in case this option is activated, all texture images referenced by the CityGML file, are loaded into the database.
 - Import appearances, do not import texture files
 - Do not import appearances – in this case, appearance information is not stored in the database; this has big influence on import performance, because caching of gml:ids referencing to gml:LinearRing geometry objects can be omitted.

Conversion from TexturedSurface (deprecated) to Appearance

- Store in theme – the attribute ‘theme’ is mandatory for the appearance class in CityGML since version 0.4.0. ‘rgbTexture’ as default value indicates that imported appearances will be textures.

Use **Restore**, **Default** or **Save** for restoring, resetting or storing preference values.

Bounding Box

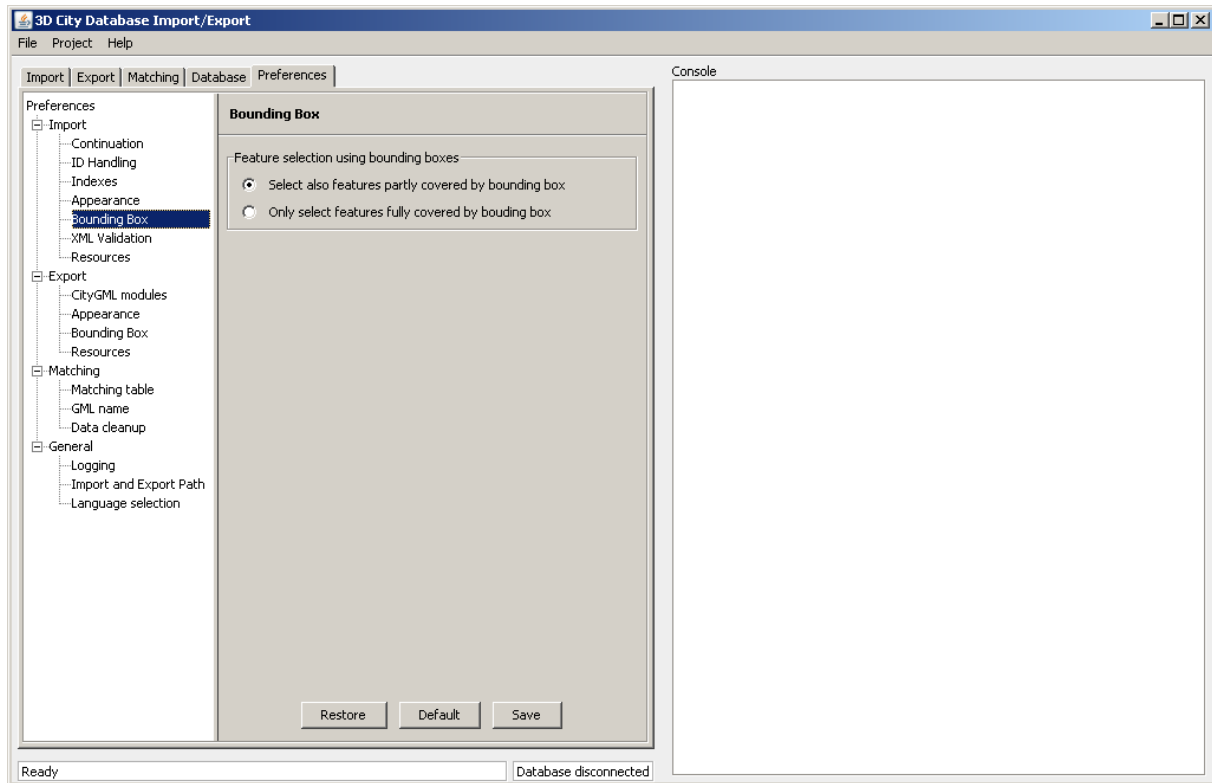


Figure 48: Import preferences – Bounding Box

If only specific areas from a dataset should be imported, bounding boxes can be used as selection criteria. Determine whether only objects fully covered should be imported or also objects whose bounding boxes are intersected by the bounding box defined in the import dialogue. Activate the radio button to decide on the import procedure.

- Feature selection using bounding boxes
 - Select also features partly covered by bounding box (default setting)
 - Only select features fully covered by bounding box

Use **Restore**, **Default** or **Save** for restoring, resetting or storing preference values.

XML Validation

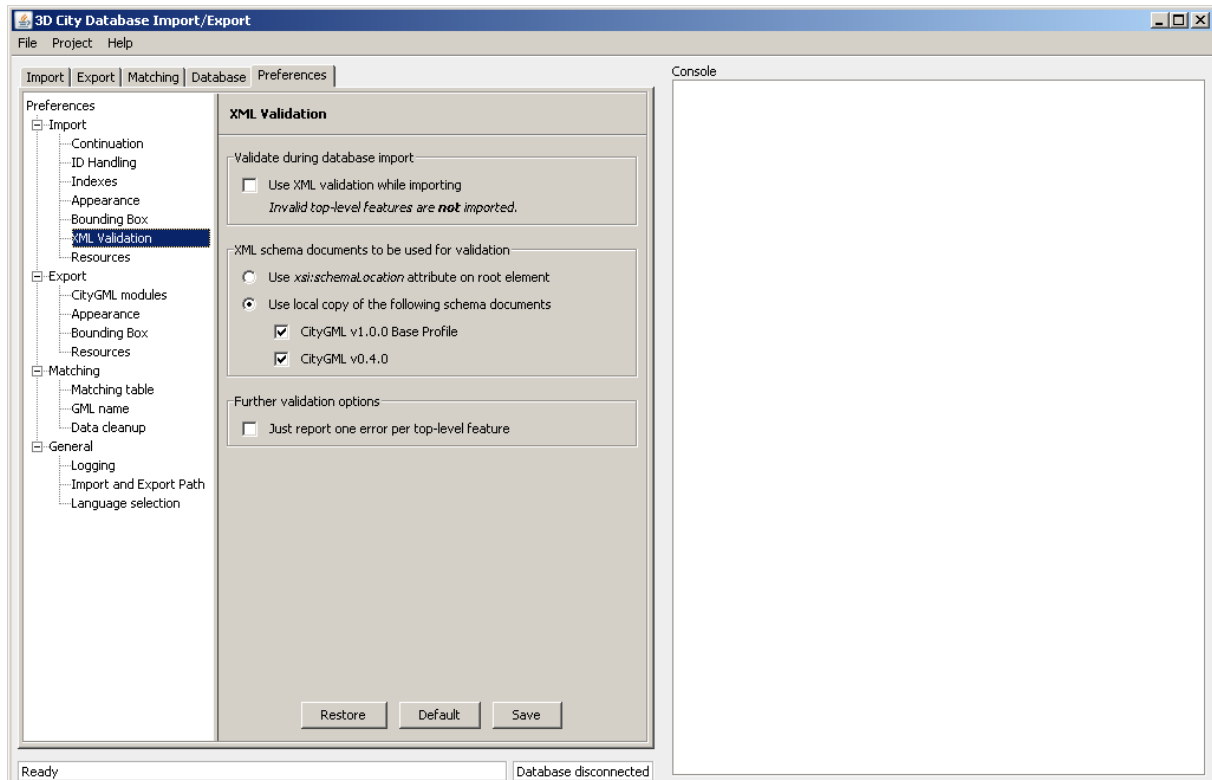


Figure 49: Import preferences – XML Validation

The Import / Export tool allows for validating CityGML instance documents against the official CityGML XML schema documents. By this means, you can ensure that the files to be imported into the database are valid with respect to the CityGML schemas. Invalid files might cause the import procedure to behave unexpectedly or even to abort abnormally. Thus, it is **strongly recommended** to only import valid CityGML instance documents as this improves the robustness of the Import / Export tool.

XML validation is performed on the level of single CityGML top level features. The instance document is parsed chunk-wise instead of processing the whole document at once as with other XML validation tools. Accordingly, this reduces main memory consumption and allows for validating documents of arbitrary file size.

Please note, that the validation functionality is explicitly tailored to the specific needs of the Import / Export tool. The tool does not provide an all-purpose XML validator. In contrast, validation is only performed for XML content associated with the official CityGML namespaces. All other XML constructs within the document are skipped and not evaluated.

The XML validation process may either be started manually by pressing the **Just validate** button on the import panel (cf. Figure 42) or automatically for each input file during database import.

- Validate during database import
 - Use XML validation while importing

If this option is checked each and every input file will be validated against the CityGML schema documents during the import process. CityGML top level features containing invalid XML content with respect to the CityGML schemas are **not imported**. Although this option generally improves the robustness of the import procedure it is not set by default because XML validation is both memory and time consuming.
- XML schema documents to be used for validation
 - Use *xsi:schemaLocation* attribute on root element

If this option is chosen the *xsi:schemaLocation* attribute on the root element of the input document is searched for the XML schema documents to be used for validation. This is neither the default option nor is it recommended to generally activate this option because it cannot be guaranteed that the input document actually points to the official CityGML schemas.

However, in case the instance document contains XML content from a CityGML Application Domain Extension (ADE) this is the only way to correctly validate the document since ADE content is specified in its own XML schema file.
 - Use local copy of the following schema documents
 - CityGML v1.0.0 Base Profile
 - CityGML v0.4.0

The Import / Export tool provides local copies of the official CityGML schemas which are used for validation by default. Local copies allow for XML validation of input documents even if a connection to the internet, and thus to the official OGC schema repository, is not available.
- Further validation options
 - Just report one error per toplevel feature

This option strongly reduces the output of error messages in the console window for invalid top level features.

Use **Restore**, **Default** or **Save** for restoring, resetting or storing preference values.

Resources

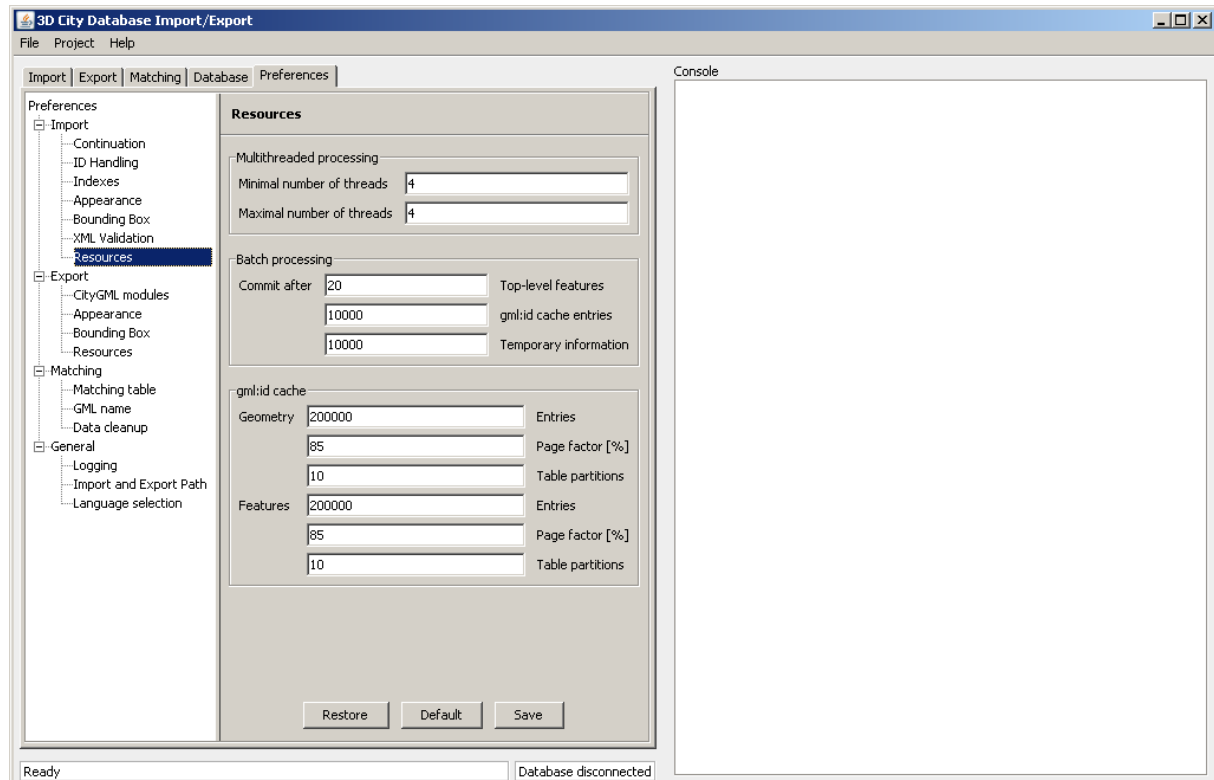


Figure 50: Import preferences – Resources

The overall architecture of the Import / Export tool is based on multithreading, i.e., concurrent execution of multiple interacting computational tasks [Stadler et al., 2009]. Generally, each task within the import/export process of CityGML data is carried out by separate threads. The decoupling of compute bound from I/O bound tasks and their parallel non-blocking processing usually leads to an increase of the overall application performance. For example, the parsing of the input document and the processing of CityGML objects is not blocked by threads waiting for database response. In a multi-core environment, threads can even be executed simultaneously on multiple CPUs or cores. The resource settings allow for controlling the number of concurrent threads used during the import process.

Furthermore, the Import / Export tool employs strategies for the handling of CityGML instance documents of arbitrary file size and the resolving of XLink references. Since backwards references are possible in CityGML instance documents, resolving of XLinks using a one-stage approach would require the whole document to be present in memory which strongly limits the maximum file size of input documents.

For this reason, the import process follows a two-phase strategy: In the first run features are written to the database neglecting references to remote objects. However, if a feature contains an XLink, any contextual information about this XLink is written to temporary database tables. This information comprises, for example, the table name and primary key of the referencing feature/geometry instance as well as the *gml:id* of the target object.

Furthermore, while parsing the document the import process keeps track of every encountered *gml:id* as well as the table name and primary key of the corresponding object in database. It is important to record this information because a priori it cannot be predicted whether or not a *gml:id* is referenced by an XLink from somewhere else in the document. In order to ensure fast access, the information is cached in memory. If the maximum cache size is reached the cache is paged to temporary database tables to prevent memory overflows.

In a consecutive run, the temporary tables containing the contextual information about XLinks are revisited and queried. Since the entire CityGML document has been processed at this point in time, valid references can be resolved and processed accordingly. With the help of the *gml:id* cache the referenced objects can be quickly identified within the database. The caching and paging behaviour can be influenced via the resource preferences.

Finally, in order to optimize database response times, multiple database operations are submitted to the database in a single request (batch processing). This allows for an efficient data processing on the database side. The number of SQL statements in one batch can be influenced by the user through the resource settings.

- Multithreaded processing

- Minimal / Maximal number of threads

Enter reasonable values for the number of concurrent threads depending on your hardware configuration. By default, each field is set to the number of available CPUs/cores times two. Before starting the import process the minimal number of threads is created. Further threads up to the specified maximal number are only created if necessary.

Please note, that a higher number of threads does not necessarily result in a better performance. In contrast, a too high number of active threads faces disadvantages such as thread life-cycle overhead and resource thrashing. For example, creating too many threads in one Java virtual machine can cause a run out of memory or thrashing due to excessive memory consumption.

- Batch Processing

The following fields influence the batch size used in batch processing. Please note, that all database operations of one batch are buffered in memory before submitting the batch to the database. Thus, the Import / Export tool might run out of memory if the batch size is too high. After each batch the transaction is committed.

- Commit after

- Top level features

Default value is 20.

- *gml:id* Cache entries

Default value is 10,000.

- Temporary information

Used for storing contextual information about XLinks. Default value is 10,000.

Be aware that these data is stored in temporary tablespace managed by Oracles DBMS which must be big enough (extend with command ALTER TABLESPACE if necessary).

- gml:id cache – used for XLink resolving
 - Geometry / Features
 - Entries

The maximum number of gml:id cache entries to be kept in memory. If this number is too high the Import / Export tool might run out of memory. Default value is 200,000.
 - Page factor %

The page factor is correlated with the setting for the number of gml:id cache entries. The page factor defines the percentage of cache entries which are paged to the database if the maximum cache size is reached (85% is the default value).
 - Table partitions

Number of temporary tables which are used in paging gml:id entries – partitioning is based on hashing the gml:ids. Default value is 10.

Use **Restore**, **Default** or **Save** for restoring, resetting or storing preference values.

4.1.2.4.2 Export preferences

CityGML modules

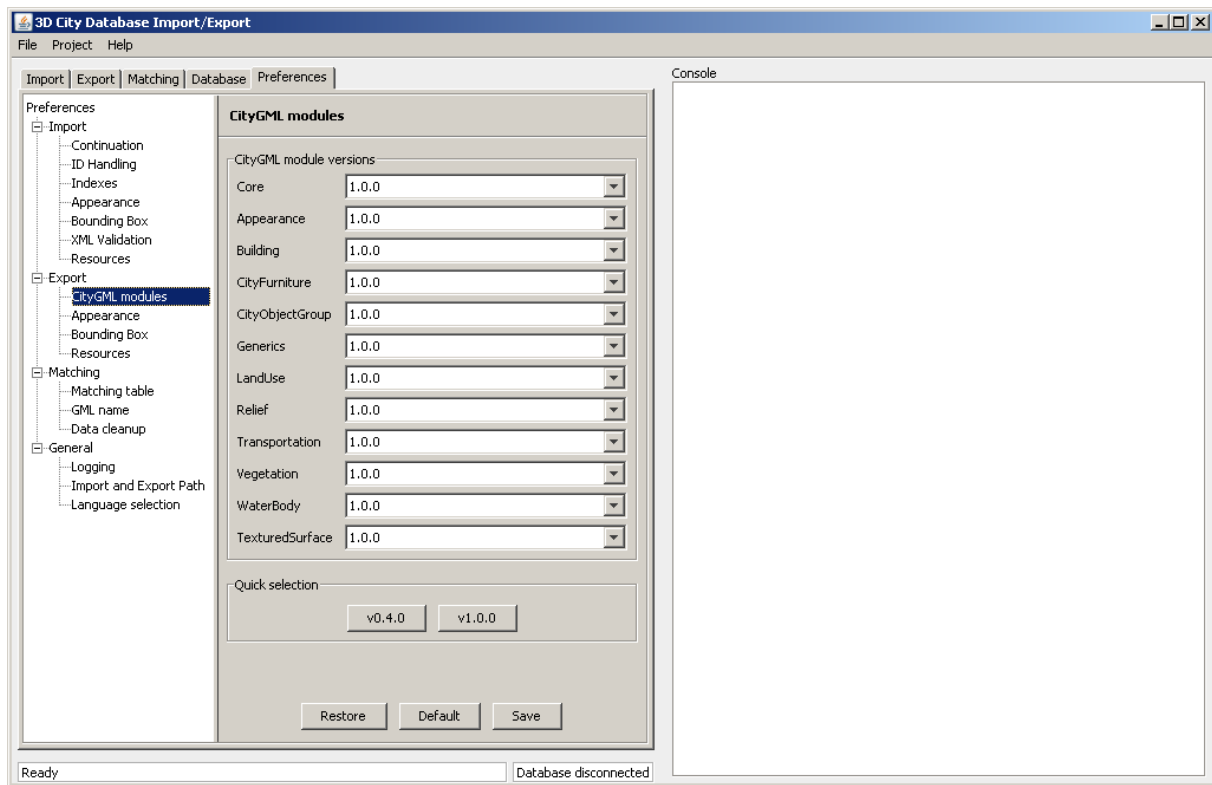


Figure 51: Export preferences – CityGML modules

The Export Tool allows exporting data in CityGML format version 0.4.0 or 1.0.0. The settings can be handled for each top-level feature class individually or jointly with just one click on the appropriate Quick selection button.

- **CityGML module versions**
Selection box for each top-level class – Choose the desired version of CityGML
Proposed value is 1.0.0.
- **Quick selection**
Choose the appropriate version and change the settings for all CityGML modules at once.

Use **Restore**, **Default** or **Save** for restoring, resetting or storing preference values.

Appearance

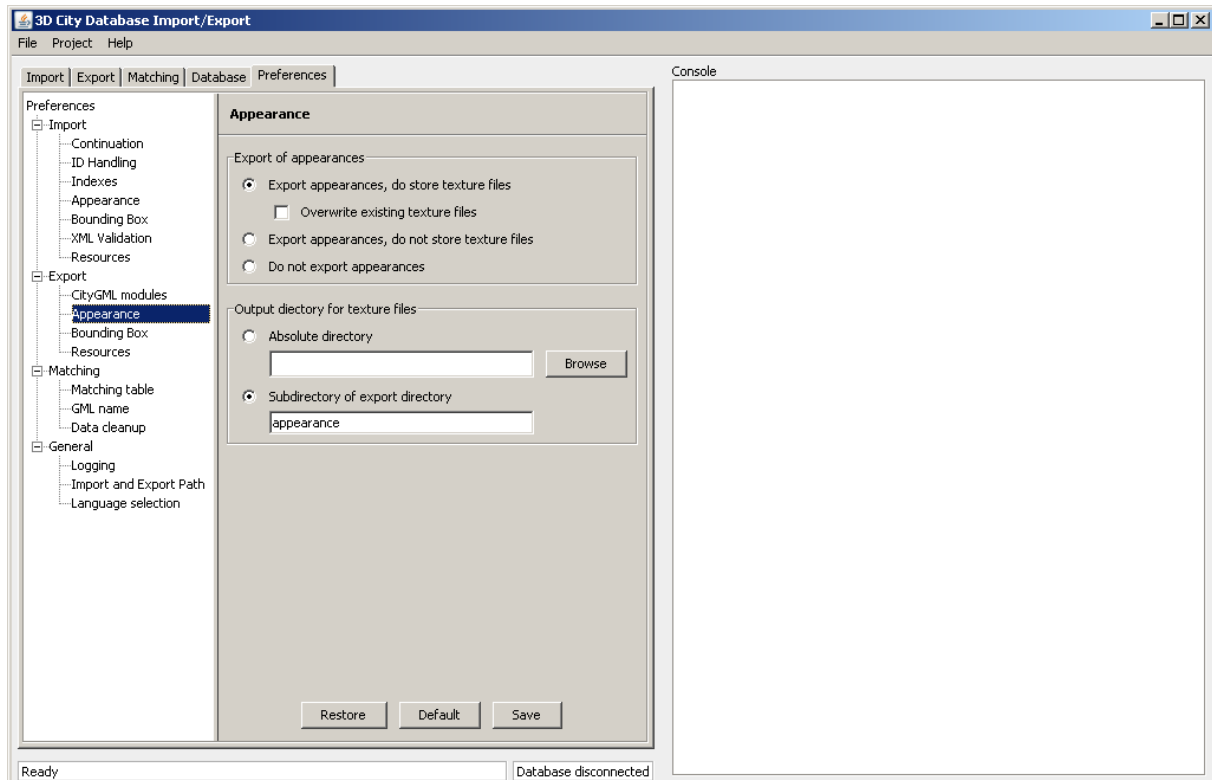


Figure 52: Export preferences – Appearance

Usage of textures is handled in a similar way compared to the import procedure; the preferences distinguish between three alternatives:

- Export of appearances
 - Export appearances, do store texture file (default setting)
 - Overwrite existing texture file
 - Export appearances, do not store texture files
 - Do not export appearances

Define where to store exported data:

- Output directory for texture files
 - Absolute directory: identify the folder using the complete name of the directory from the / (root) directory downward (absolute path, e.g. for MS Windows D:\directory\output_directory or UNIX/LINUX: /root/output_directory).
 - Subdirectory of export directory (default setting): insert the subdirectory name only, without specifying the complete path (proposed value is appearance).

Use **Restore**, **Default** or **Save** for restoring, resetting or storing preference values.

Bounding Box

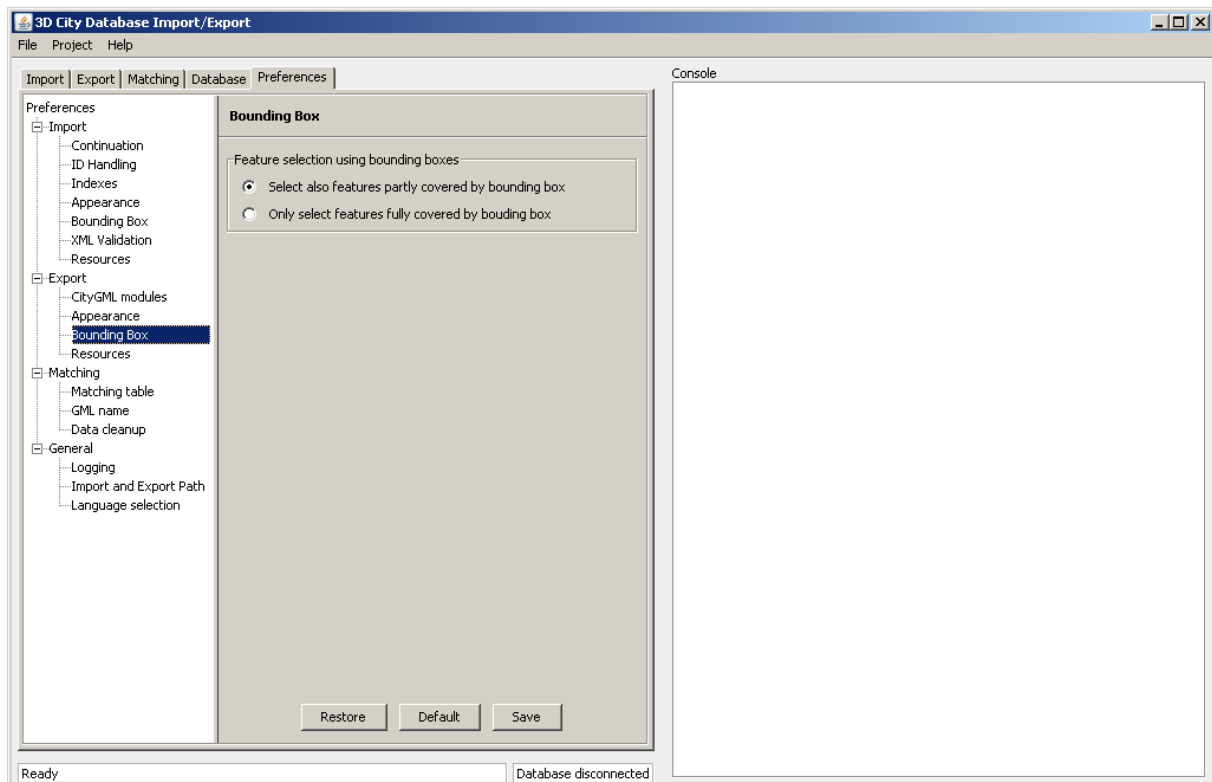


Figure 53: Export preferences – Bounding Box

This dialogue box is similar to the import window displayed in Figure 48. Activate the radio button to determine how to handle objects whose bounding boxes are intersected by the bounding box defined in the Export tab.

- Feature selection using bounding boxes
 - Select also features partly covered by bounding box (default setting)
 - Only select features fully covered by bounding box

Use **Restore**, **Default** or **Save** for restoring, resetting or storing preference values.

Resources

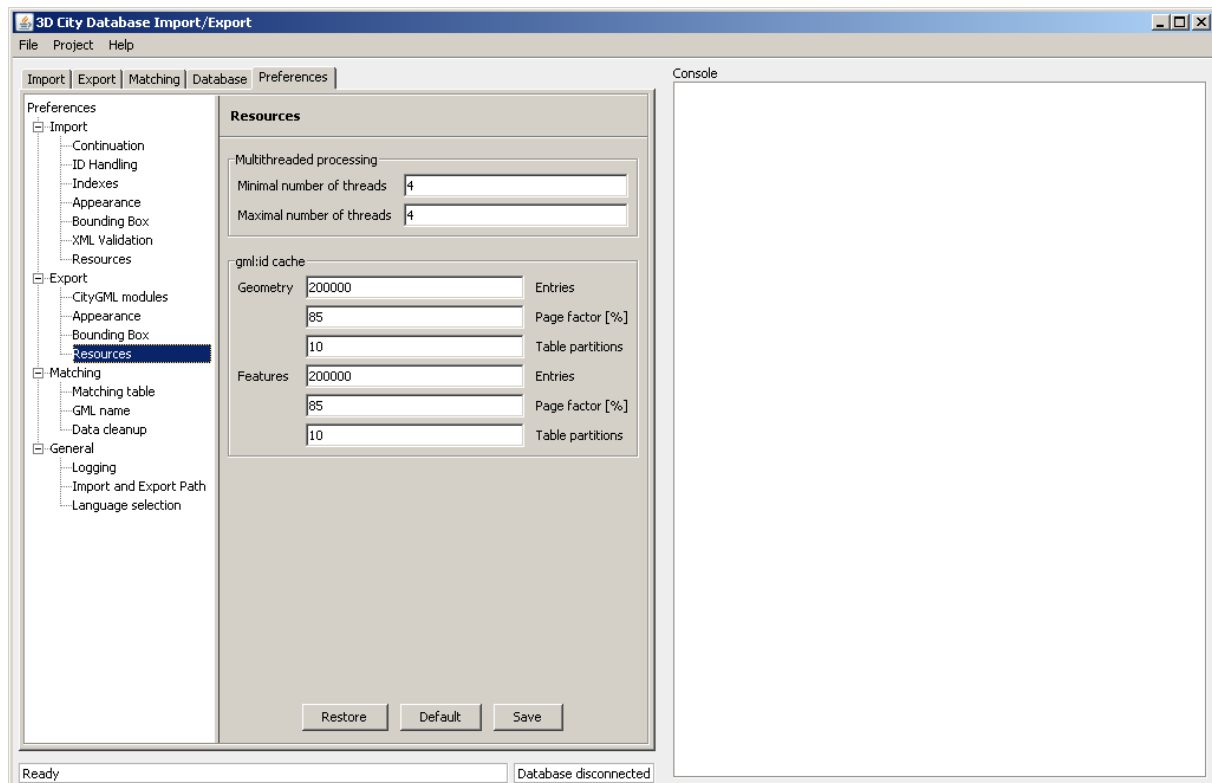


Figure 54: Export preferences – Resources

Just like with the import procedure (cf. Figure 50), the export process is based on multithreaded processing in order to increase the application performance. The resource settings allow for controlling the number of concurrent threads used during the export process.

In order to rebuild XLink references the Import / Export tool has to keep track of each and every *gml:id* of exported features and geometry objects. Prior to exporting an object it is checked whether an object with the same *gml:id* has already been processed. For this purpose, the *gml:ids* are kept in a main memory cache to allow fast access. If the predefined cache size limit is reached the cache is paged to temporary database tables. The caching and paging behaviour can be influenced via the resource preferences.

- Multithreaded processing
 - Minimal / Maximal number of threads

Enter reasonable values for the number of concurrent threads depending on your hardware configuration. By default, each field is set to the number of available CPUs/cores times two. Before starting the export process the minimal number of threads is created. Further threads up to the specified maximal number are only created if necessary.

Please note, that a higher number of threads does not necessarily result in a better performance. In contrast, a too high number of active threads faces disadvantages such as thread life-cycle overhead and resource thrashing. For example, creating too many threads in one Java virtual machine can cause a run out of memory or thrashing due to excessive memory consumption.

- gml:id cache – used for XLink resolving
 - Geometry / Features
 - Entries

The maximum number of gml:id cache entries to be kept in memory. If this number is too high the Import / Export tool might run out of memory. Default value is 200,000.
 - Page factor %

The page factor is correlated with the setting for the number of gml:id cache entries. The page factor defines the percentage of cache entries which are paged to the database if the maximum cache size is reached (85% is the default value).
 - Table partitions

Number of temporary tables which are used in paging gml:id entries – partitioning is based on hashing the gml:ids. Default value is 10.

Use **Restore**, **Default** or **Save** for restoring, resetting or storing preference values.

4.1.2.4.3 General preferences

Logging

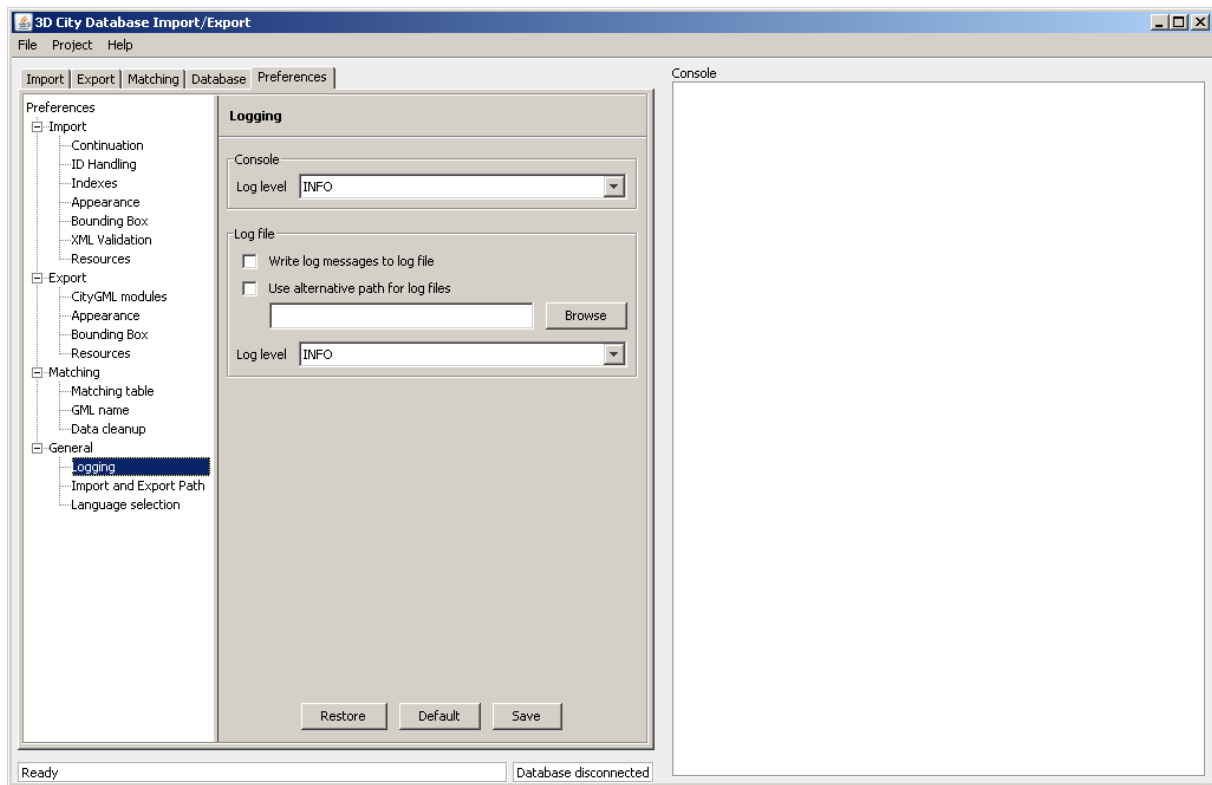


Figure 55: General Preferences – Logging

Log messages are used to record information about events such as activities or failures, e.g. during import and export procedures. The messages include information about the time of the event as well as its severity. Log messages are always printed to the console window and may additionally be forwarded to a log file on your local computer.

The following four log levels are distinguished (from highest severity to lowest severity):

- **ERROR** – an error has occurred in the program (usually an exception). This comprises internal and unexpected failures. Furthermore, the XML validation process reports invalid XML content of CityGML instance documents via this log level.
- **WARN** – an anomalous condition has been detected and the program will attempt to deal with it.
- **INFO** – an interesting piece of information that helps to give context to the log, often when processes are starting or stopping.
- **DEBUG** – additional messages reporting the internal state of the program.

Please note, that the logging output in the console window will be automatically truncated after 10,000 log messages in order to prevent high main memory consumption.

- Console
 - Log level

Sets the minimum log level for messages printed to the console window. In particular, a log record whose severity is greater or equal to this log level is printed (default: INFO).
- Log file
 - Write log messages to log file

Activate this option if the log messages shall be printed to a local log file in addition to the console. The log file name is automatically set as "log_3dcitydb_impexp_<date>.log" where <date> is replaced with the current date at program startup. If the log file does not exist already it is created by the Import / Export tool. Otherwise, log messages are appended to an existing log file.

Please note, that the Import / Export tool does not provide any kind of log file rotation mechanism.
 - Use alternative path for log files

By default, log files are stored within the subfolder "log" of the installation directory. This option allows for specifying an alternative folder.
 - Log level

Sets the minimum log level for messages printed to the log file. In particular, a log record whose severity is greater or equal to this log level is printed (default: INFO).

Use **Restore**, **Default** or **Save** for restoring, resetting, or storing preference values.

Import and Export Path

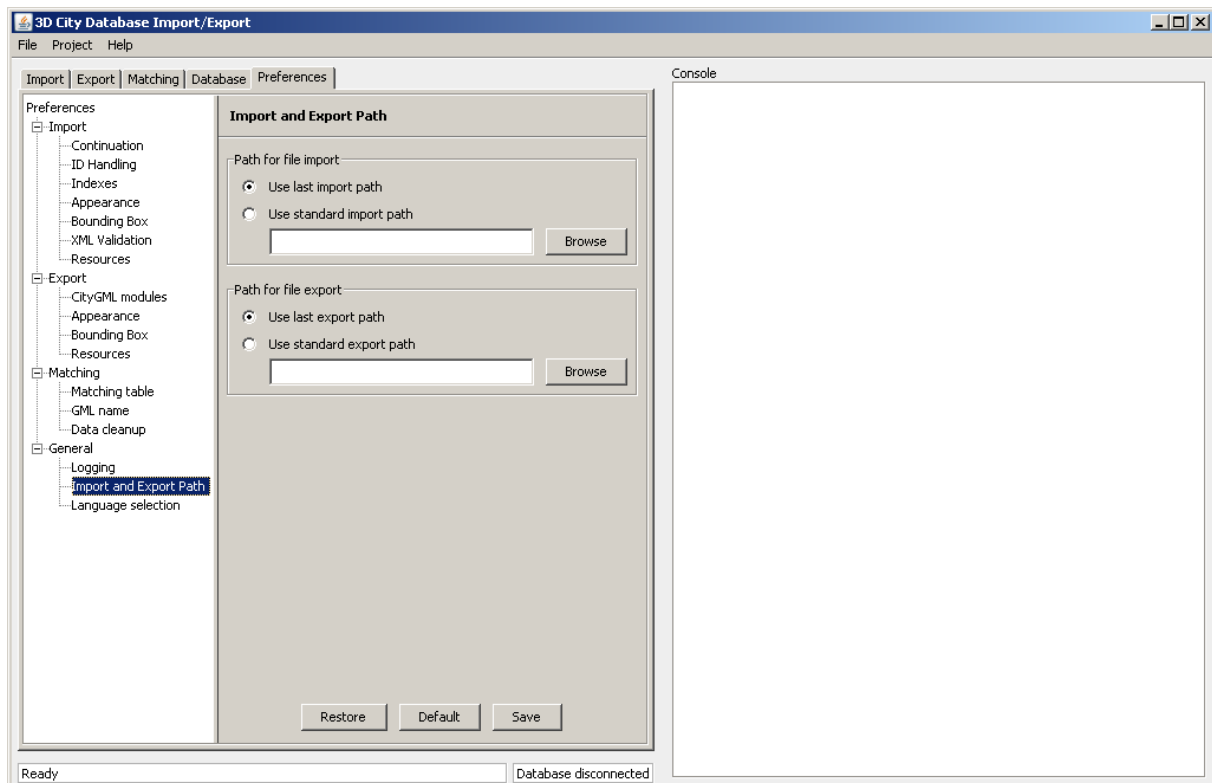


Figure 56: General Preferences – Import and Export Path

Enter the path for file import or export. Choose between the last used import / export path, or browse for a specific directory. The selected directory will be set as default in all subsequent dialogues that require an input / output file.

- Path for file import
 - Use last import path (default setting)
 - Use standard import path
- Path for file export
 - Use last export path (default setting)
 - Use standard export path

Use **Restore**, **Default** or **Save** for restoring, resetting, or storing preference values.

Language selection

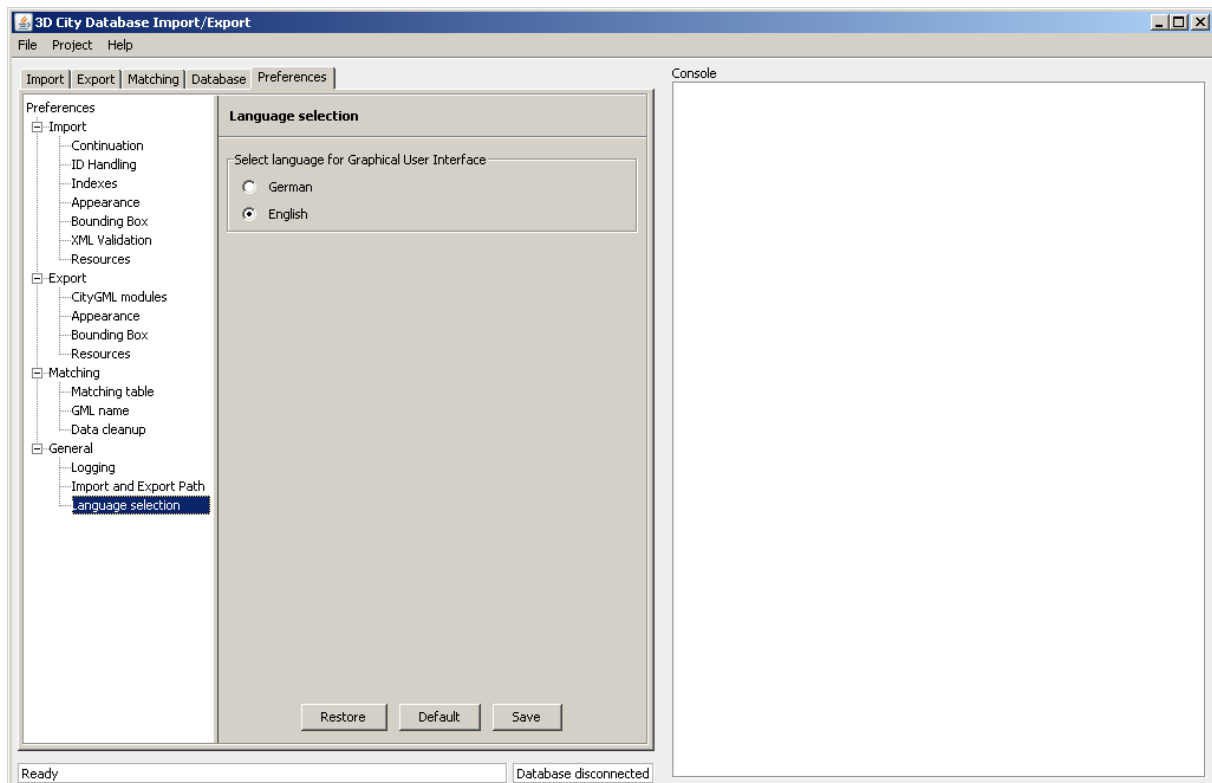


Figure 57: General preferences – Language selection

The Import / Export Tool's GUI is available in German and English language. Use one of the radio buttons for adjusting to your preferred language.

- Language selection
 - German
 - English

Use **Restore**, **Default** or **Save** for restoring, resetting, or storing preference values.

5 Matching Tool

5.1 Motivation

The 3D city database is used as a data management component, where geo related information is collected, updated, and exchanged. Various providers of 3D city models feed their data into the centralised database. Therefore, CityGML as standardized data format is used to ensure interoperable data structuring. But this is not enough. All too often reality looks like this: city objects are modelled by various providers based on different data sources and in different levels of detail. When transferring several of these models into the database, data consistency is not guaranteed any more. Since sharing of data avoids duplication of work, a solution is needed to harmonise different models in order to end up with one consistent city model.

5.2 Idea

If at least two models of the same object are on-hand, one might ask which of them is the “better one”, or “more trusted one”, i.e., which one forms the basis in the harmonization process. Given its legal background, a state-controlled model has to remain unchanged in terms of its initial geometry and semantics and has to be used as the reference dataset. Despite the fact, that another model probably is more accurate and detailed, it has to be merged into the reference dataset. As a result, a harmonized city model is now available containing not only the official geometry but also the higher detailed representations used for advanced visualisation and analysis. The two datasets are distinguished by their LINEAGE attribute which is specified before data import (see figure 44 for the respective GUI component).

5.3 Approach

Following steps are carried out to find and merge corresponding building models:

- **Compare building positions**

Each 3D building is flattened to a 2D footprint by neglecting its height information (one LoD is chosen to perform the flattening). Footprints of reference and buildings to fuse are compared to find candidate pairs, which overlap to some degree.

- **Find corresponding building**

Minimum overlapping percentages for both directions ($(\text{area_ref} \cap \text{area_merge}) / \text{area_ref}$ and $(\text{area_merge} \cap \text{area_ref}) / \text{area_merge}$, cf. Figure 58) are used to determine corresponding building pairs. The percentage values are controlled by settings in the user interface.

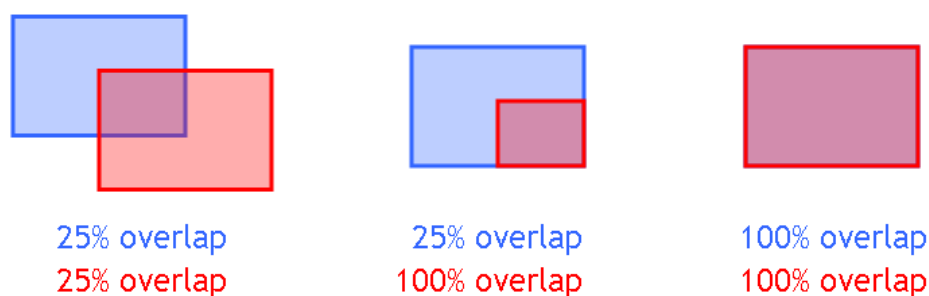


Figure 58: The overlap of two objects can be defined by two percentage values

Merging geometry

Moreover, source and target LoD need to be defined for the fusion process. According to the specified source LoD, geometry has to be collected from various locations in the building model. Figure 59 gives an overview of the building model's class complexity subject to different LoDs (1 to 4). This yields following target classes for geometry allocation:

- ≥ LOD1: BUILDING (to retrieve all elements of class `_AbstractBuilding`, including Buildings as well as BuildingParts)
- ≥ LOD2: Building_Installation (exterior)Thematic_Surface
- ≥ LOD3: Opening
- ≥ LOD4: Room
Building_Furniture,
Building_Installation (part of rooms, interior)
Thematic_Surface (part of rooms)
Opening (part of rooms)

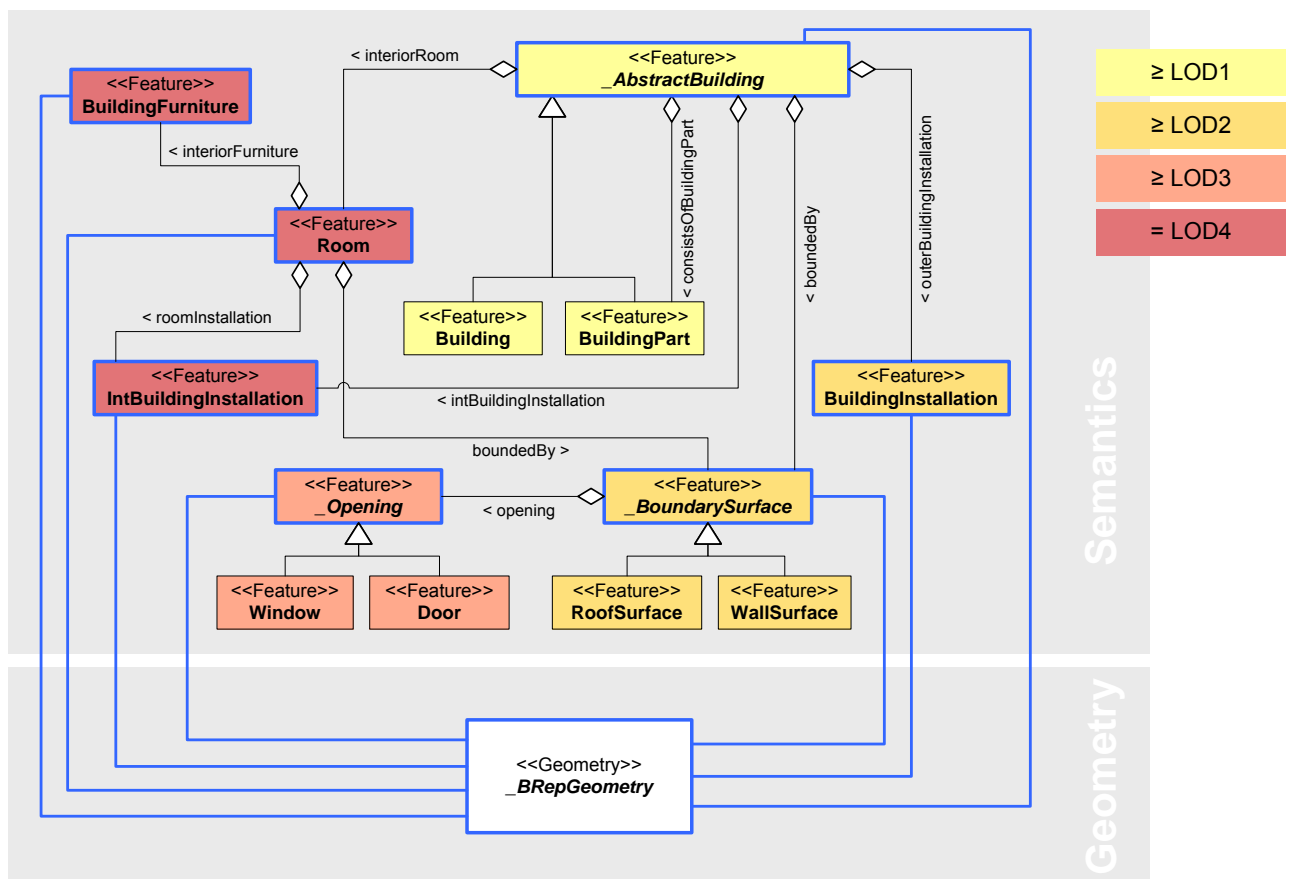


Figure 59: The complexity of the building model varies with the use of different LODs

It is very likely, that the two building datasets have different semantic structures. When aiming at a consistent building dataset, only one of these structures can be maintained. If cadastral datasets are available use those first. The building geometry of a second, geometrically more precise dataset must be transferred directly into the main building feature of the first object. While the geometric structure persists, semantic structuring is neglected. All multisurfaces that contain the more precise geometry of the various semantic objects are

joined into a single multisurface for the whole building. This flattening is required, since a GML multisurface is incapable of containing other multisurfaces by design and the buildings allows for at most one multigeometry object per level of detail. Figure 60 provides an excerpt from the GML class hierarchy to illustrate this restriction. A GML multisurface can contain surfaceMembers of type `_Surface`. Since `MultiSurface` is not derived from `_Surface`, nested multisurfaces are disallowed (in contrast to `CompositeSurfaces`, which can be nested).

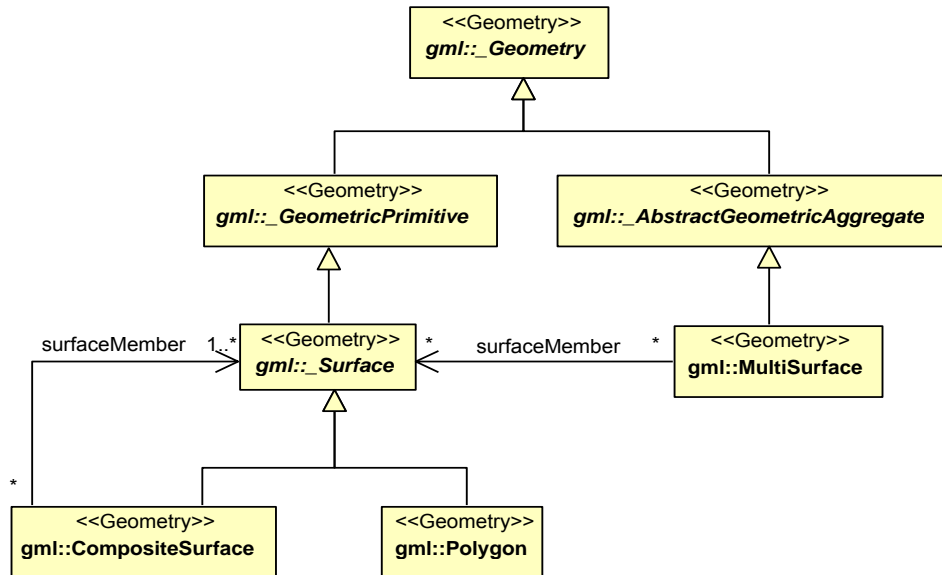


Figure 60: GML surface geometry classes (excerpt)

5.4 Graphical user interface

The matching process is separated into two consecutive steps:

- Step 1: Compute building overlaps
- Step 2: Transfer geometry of corresponding buildings.

For both steps, you are asked to define parameters. Start the process by clicking on the respective start button. Figure 61 shows the matching tool's user interface and explains the main alternatives of parameter adjustment.

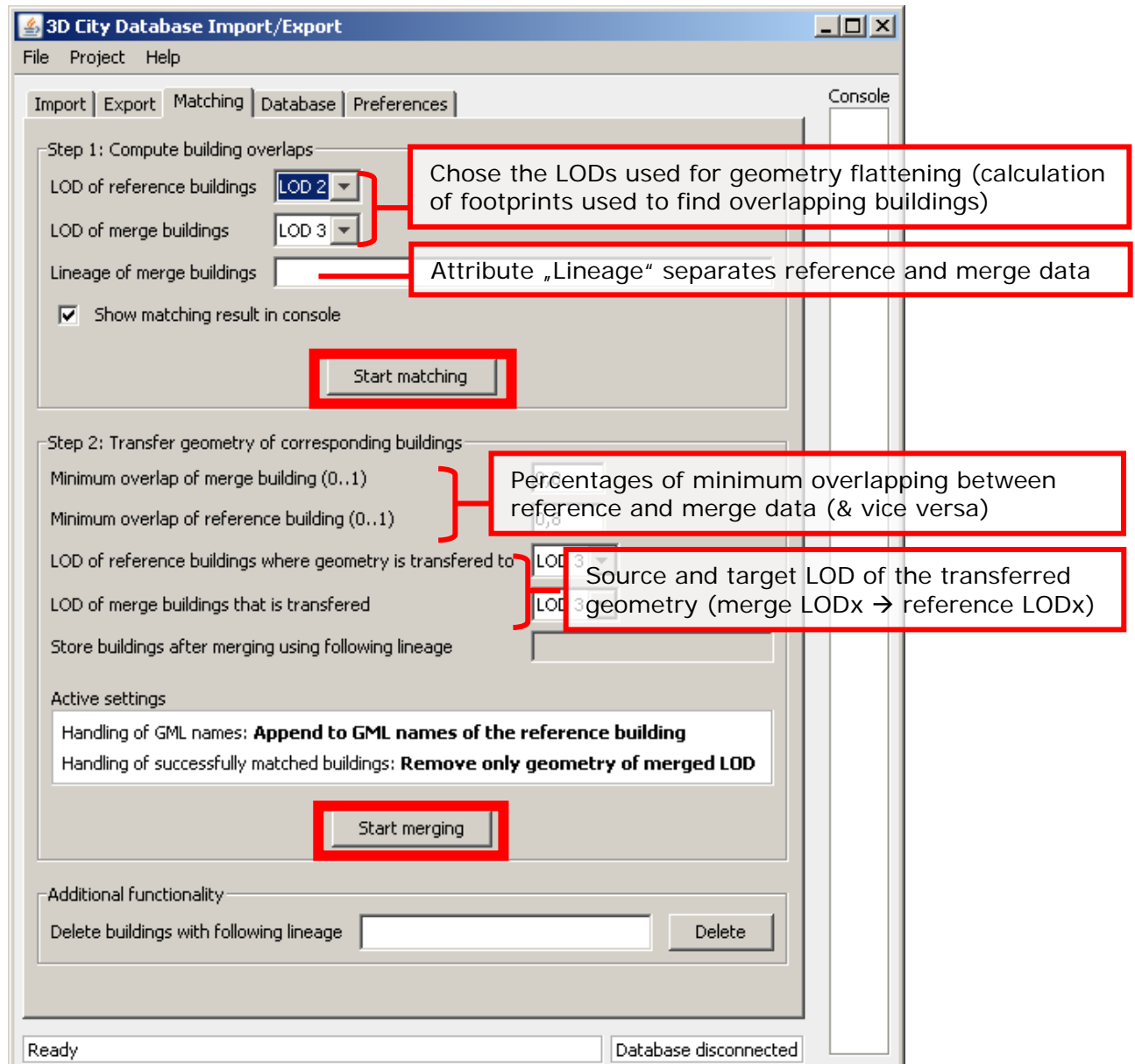


Figure 61: User interface of the matching tool

The blue text in Figure 61 gives a brief overview of matching setting from the preferences tab. There are three option panels which are explained below.

Matching table

The matching table contains all pairs of buildings with overlapping footprints. To get an impression of the correspondencies within the data, the matching table may be plotted to the console (cf. the respective check box in Figure 61). In the preferences tab, the user may choose the number of presented rows (cf. Figure 62). Be aware that selecting the last option “Show the whole table” may cause long waiting periods.

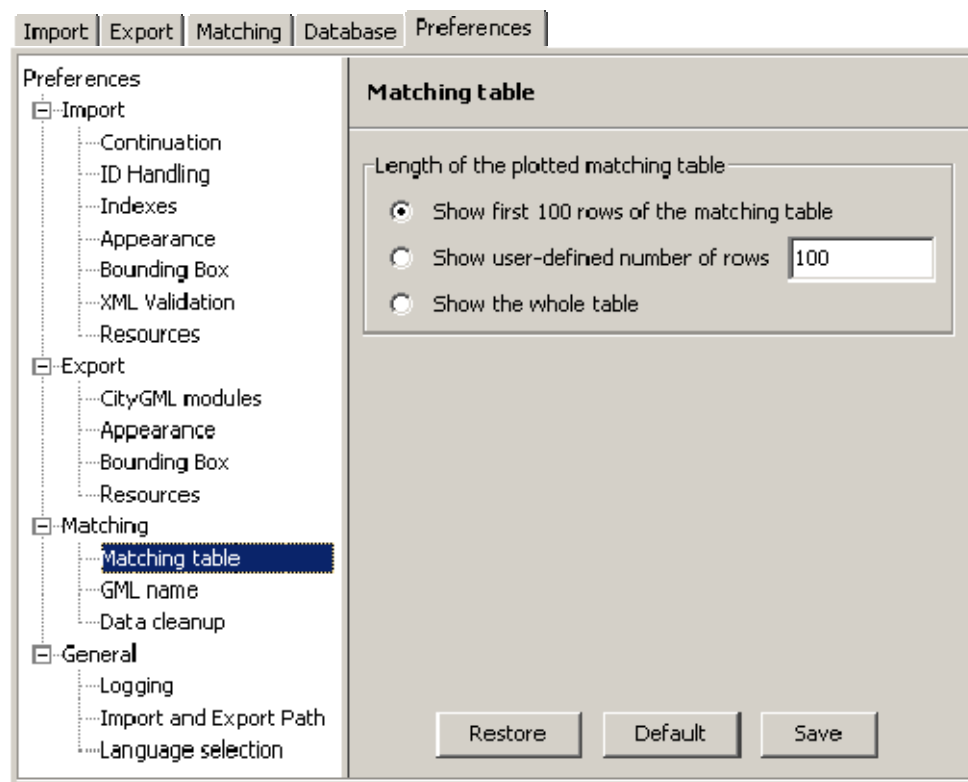


Figure 62: Options for the matching table output length

GML name

There might be inconsistencies between GML names (attribute gml:name) from the reference and the fusion data set. Therefore, different methods are possible to handle GML names from the merge data set. They might be appended to existing GML names from the reference data set, they might be ignored, or they might even overwrite GML names from the reference data set (cf. Figure 63).

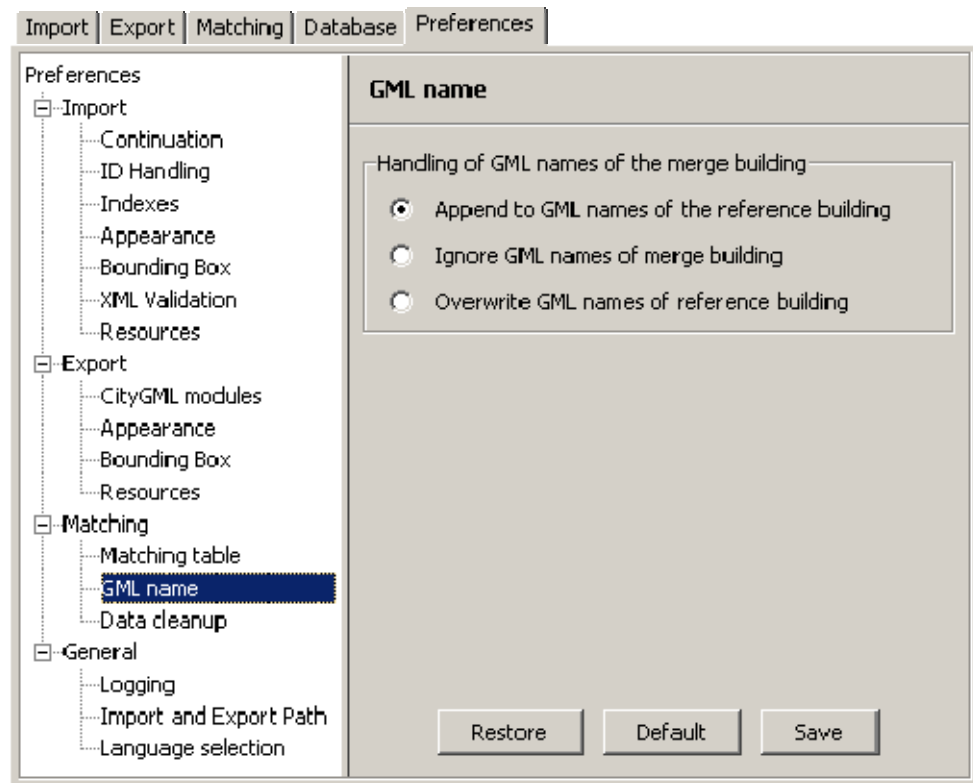


Figure 63: Options for handling of GML names from the merge data set.

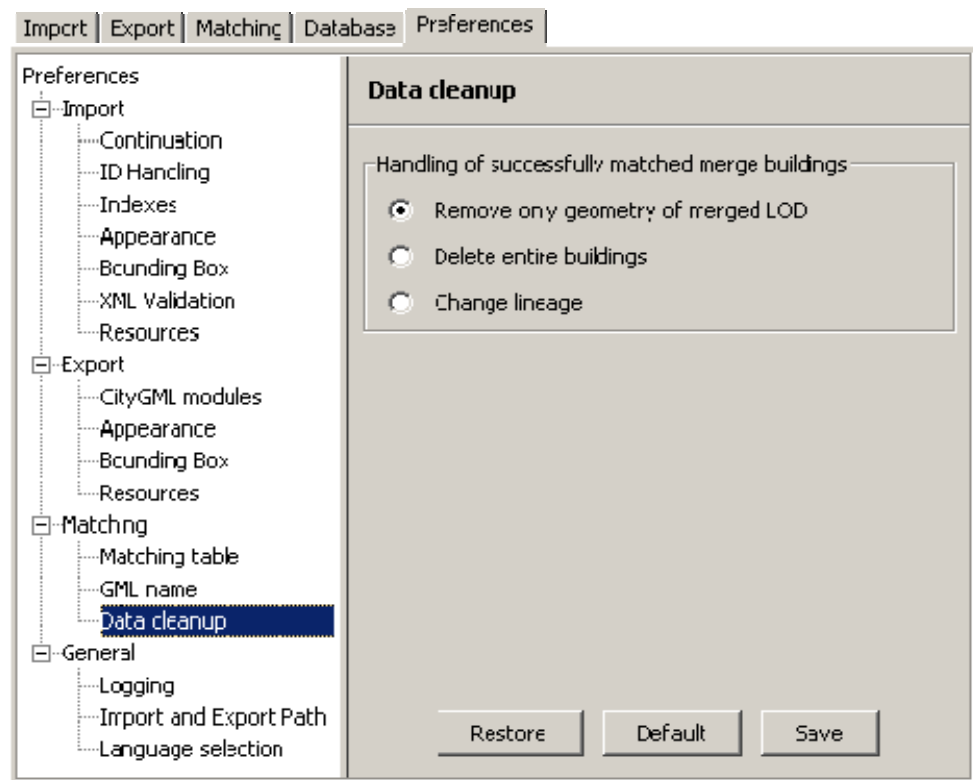


Figure 64: Options for data cleanup.

Data cleanup

Finally, three options for data cleanup are available (cf. Figure 64). If a building has been fused successfully, the geometry of the merged LoD will be removed in any case. Additionally, the building might be deleted entirely, or its lineage may be changed to a specific string value (which has to be specified in the respective text field in Figure 61). Using this last option, further LoDs may be merged in a second step.

5.5 Realisation: Stored PL/SQL Procedures

5.5.1 Matching

Find buildings with overlapping footprints and calculate their intersections

5.5.1.1 create_matching_table

Main procedure that produces the table MATCH_RESULT. It relies on the procedures and functions described in the following sections. However, it is sufficient to only execute this procedure in order to run the matching process as it internally invokes all the other functions and procedures.

Syntax:

```
procedure create_matching_table (
    lod_cand          number,
    lineage           cityobject.lineage%type,
    lod_master        number,
    tolerance         number,
    aggregate_building number);
```

Parameters:

lod_cand	LoD of the candidate geometry, $1 \leq \text{lod_cand} \leq 4$
lineage	Description of data origin. Corresponds to the attribute LINEAGE of the CITYOBJECT table. This parameter is used to distinguish the candidate buildings from the reference (master) data. Candidate buildings are those whose LINEAGE attribute values are equal to this input parameter. Reference buildings are the other building instances.
lod_master	LoD of the master geometry, $1 \leq \text{lod_cand} \leq 4$
tolerance	This value is used for the correspondent tolerance input parameter of the built-in spatial operations of ORACLE 10G R2.
aggregate_building	If set to "1" a single footprint will be calculated from the geometries of the building and all its associated building parts. If set to "0" separate footprints are calculated (default = 1).

5.5.1.2 allocate_cand_building

Retrieve IDs and geometry of LoD *lod* from all buildings with the specified *lineage*. Results are stored in a temporary table

Syntax:

```
procedure allocate_cand_building (
    lod_          number,
    lineage       cityobject.lineage%type);
```

Parameters:

lod	Level of detail (LoD) of the candidate geometry, $1 \leq lod \leq 4$
lineage	Description of data origin. Corresponds to the attribute LINEAGE of the CITYOBJECT table. This parameter is used to distinguish the candidate buildings from the reference (master) data. Candidate buildings are those whose LINEAGE attribute values are equal to this input parameter. Reference buildings are the other building instances.

5.5.1.3 allocate_geometry

Collect all geometry of all buildings listed in a temporary table into a new temporary table For $lod > 1$, also consider related thematic surfaces and external building installations

Syntax:

```
procedure allocate_geometry (
    lod_          number);
```

Parameters:

lod	Level of detail (LoD) of geometry objects, $1 \leq lod \leq 4$
-----	----------------------------------------------------------------

5.5.1.4 rectify_geometry

Repair the collected geometry and delete degenerates.

Syntax:

```
procedure rectify_geometry (
    tolerance     number);
```

Parameters:

tolerance	This value is used for the correspondent tolerance input parameter of the built-in spatial operations of ORACLE 10G R2.
-----------	-------------------------------------------------------------------------------------------------------------------------

5.5.1.5 aggregate_geometry

Join each building's geometry (using the UNION operator) from a temporary table and store it in table *table_name*.

Syntax:

```
procedure aggregate_geometry (
    table_name          varchar2,
    tolerance           number
    aggregate_building  number);
```

Parameters:

table_name	Name of the temporary table where the aggregated geometries shall be stored. For candidate geometries choose MATCH_CAND_AGGR_GEOM as table name and for master geometries choose MATCH_MASTER_AGGR_GEOM.
tolerance	This value is used for the correspondent tolerance input parameter of the built-in spatial operations of ORACLE 10G R2.
aggregate_building	If set to "1" a single footprint will be calculated from the geometries of the building and all its associated building parts. If set to "0" separate footprints are calculated (default = 1).

5.5.1.6 allocate_master_building

Retrieve IDs and geometry of LoD *lod* from all reference buildings not belonging to the specified *lineage*. The building must interact with the minimum bounding rectangle of all merge buildings. Results are stored in a temporary table.

Syntax:

```
procedure allocate_master_building (
    lod          number,
    lineage      cityobject.lineage%type);
```

Parameters:

lod	Level of detail (LoD) of master geometry object, $1 \leq lod \leq 4$
lineage	Description of data origin. Corresponds to the attribute LINEAGE of the CITYOBJECT table. This parameter is used to distinguish the candidate buildings from the reference (master) data. Candidate buildings are those whose LINEAGE attribute values are equal to this input parameter. Reference buildings are the other building instances.

5.5.1.7 join_cand_master

Create table MATCH_RESULT as a join of all merge and reference buildings. Additionally store footprint areas, the geometry of the footprints' intersection, the intersection's area, and ratios between the various areas. Delete all entries that have an empty intersection.

Syntax:

```
procedure join_cand_master (
    lod_cand      number,
    lineage       cityobject.lineage%type,
    lod_master    number,
    tolerance     number);
```

Parameters:

lod_cand	LoD of the candidate geometry, $1 \leq \text{lod_cand} \leq 4$
lineage	Description of data origin. Corresponds to the attribute LINEAGE of the CITYOBJECT table. This parameter is used to distinguish the candidate buildings from the reference (master) data. Candidate buildings are those whose LINEAGE attribute values are equal to this input parameter. Reference buildings are the other building instances.
lod_master	LoD of the master geometry, $1 \leq \text{lod_master} \leq 4$
tolerance	This value is used for the correspondent tolerance input parameter of the built-in spatial operations of ORACLE 10G R2.

5.5.1.8 function aggregate_mbr

Returns the minimum bounding rectangle (mbr) of all geometry contained in table *table_name*.

Syntax:

```
function aggregate_mbr (
    table_name    varchar2 );
```

Parameters:

table_name	Name of the temporary table where the aggregated minimum bounding rectagle shall be stored
------------	--------------------------------------------------------------------------------------------

Return value:

mdsys.sdo_geometry Oracle geometry representing the mbr.

5.5.1.9 aggregate_geometry_by_id

Utility function to unify all geometry belonging to the building with ID *id*. If *aggregate_building* is 1, also include all child buildings.

Syntax:

```
function aggregate_geometry_by_id (
    id                number,
    tolerance         number,
    aggregate_building number)
```

Parameters:

<i>id</i>	ID of selected building
<i>tolerance</i>	This value is used for the correspondent tolerance input parameter of the built-in spatial operations of ORACLE 10G R2.
<i>aggregate_building</i>	If set to "1" a single footprint will be calculated from the geometries of the building and all its associated building parts. If set to "0" separate footprints are calculated (default = 1).

Return value:

`mdsys.sdo_geometry` Oracle geometry for aggregated building geometry

5.5.2 Merging

Merge geometry of corresponding buildings (buildings with minimum intersections).

5.5.2.1 create_relevant_matches

Retrieve all match tuples from the matching table with more than the user-specified percentages of area coverage.

Syntax:

```
procedure create_relevant_matches (
    delta1            number,
    delta2            number);
```

Parameters:

<i>delta1</i>	Intersection of candidate and master footprint in relation to the area of the candidate footprint
<i>delta2</i>	Intersection of master and candidate footprint in relation to the area of the master footprint

5.5.2.2 collect_all_geometry

Collect geometry IDs of all relevant buildings and their sub parts and put them into a new table. According to the respective LOD the following tables have to be analysed:

≥ LOD1

BUILDING (to retrieve instances of the abstract class `_AbstractBuilding`, specializing to classes `Building` and `BuildingPart`)

≥ LOD2

BUILDING_INSTALLATION where the attribute `external` is set to 1 (to retrieve instances of the class `BuildingInstallation`),

Thematic_Surface which is part of a relevant building (to retrieve instances of the abstract class `_BoundarySurface`, specializing to classes `WallSurface`, `GroundSurface`, etc.)

≥ LOD3

OPENING which is part of a relevant thematic surface (to retrieve instances of the abstract class `_Opening`, specializing to classes `Door` and `Window`)

= LOD4

ROOM which is part of a relevant building (to retrieve instances of the class `Room`),

BUILDING_FURNITURE which is part of a relevant room (to retrieve instances of the class `BuildingFurniture`),

BUILDING_INSTALLATION which is part of a relevant room or which is part of a relevant building and where the attribute `external` is set to 0 (to retrieve instances of the class `IntBuildingInstallation`),

THEMATIC_SURFACE which is part of a relevant room (to retrieve instances of the abstract class `_BoundarySurface`, specializing to classes `WallSurface`, `GroundSurface`, etc.),

OPENING which is part of a relevant thematic surface (to retrieve instances of the abstract class `_Opening`, specializing to classes `Door` and `Window`)

Syntax:

```
procedure collect_all_geometry (
    lod          number ) ;
```

Parameters:

lod Level of detail (LoD) of candidate geometry, $1 \leq lod \leq 4$

5.5.2.3 move_appearance

Move all appearances of affected geometries to one new appearance feature in the reference building.

Syntax:

```
procedure move_appearance ( ) ;
```

Parameters:

none

5.5.2.4 create_and_put_container

Create a new geometry for each relevant building. It acts as container where sub geometries will be hooked into. According to the chosen GML name handling, the reference building's GML name will be either kept or replaced/appended by the merge building's GML name.

Syntax:

```
procedure create_and_put_container (  
    lod            number,  
    name_mode      number,  
    delimiter      varchar2);
```

Parameters:

lod	Level of detail (LoD) of geometry object, $1 \leq lod \leq 4$
name_mode	code that represents the chosen GML name handling
delimiter	

5.5.2.5 move_geometry

Change the reference of all sub geometries to the new container geometries by updating their parent and root IDs.

Syntax:

```
procedure move_geometry ();
```

Parameters:

none

5.5.2.6 delete_multi_surfaces

Delete old high level geometries which are now replaced by the container geometries.

Syntax:

```
procedure delete_multi_surfaces ();
```

Parameters:

none

5.5.2.7 cleanup

Delete all temporary tables.

Syntax:

```
procedure cleanup ();
```

Parameters:

none

6 Version and history management

6.1 Preface

Tools for version and history management could be transferred without major changes from the previous 3D city model project, developed at the University of Bonn in cooperation with the company Lat/Lon. The corresponding chapter of the old documentation was translated to English for the following paragraphs. The so called *Planning Manager* can be downloaded from 3D City Database homepage (<http://www.3dcitydb.org> – scripts & tools)

6.2 The concept of versions and CityModelAspects

The capability of the 3D geodatabase is not limited to mere representation of data sets and management of temporal changes of these data (history). Instead the data can also be used to initialise a new planning processes for certain city areas. These planning projects are characterized by the fact that different concepts and models (versions) may exist for the same spatial area. The database schema introduced in this chapter allows to define geographically constrained planning areas and to assign an arbitrary number of planning alternatives to each planning area.

For a better understanding the benefits and the necessity of version and history management consider the following example:

At the time t , it is decided to carry out certain replacements in the city districts X and Y. Consequently, spatially restricted planning areas are defined. The city planners A and B develop different concepts for the area X. The city planners I and II do so for the area Y. All of their plannings are based on the same basic data set of the time t . Every city planner will delete or change existing buildings and add new buildings. Partly, the city planners A and B or I and II will change the same building in the basic data set.

For the presentation of partial results or coordination between the concepts for X and Y it is necessary, for example to combine the plannings of A and B and to present them in one 3D city model.

In addition, it can be necessary to reset the planning of the city planner B to the, meanwhile overwritten, situation of the time t_1 or to look at the whole state of the database including all plannings once again from the situation at the (past) time t_2 .

All scenarios described in the example can be realized using the database schema presented in this chapter. The Workspace Manager, provided by the Oracle 10g_R2 DBMS and higher, is used for history management and versioning of the data².

Figure 65 shows the hierarchy of parallel plannings and highlights the names used in the following:

- **LIVE** is the original data set (*GeneralWorkspace*) forming the basis for all plannings.
- **Plannings** (*Planning*) are fixed spatial areas that are subject to a planning decision.
- **Planning alternatives** (*PlanningAlternative*) are (arbitrary many) parallel existing models of the changes that are associated to one planning.

² At this point, the official manual for the Oracle Workspace manager has to be mentioned. It can be found under ID B10824-01 with the title "Application Developer's Guide - Workspace Manager" on the web page of Oracle Inc. It offers extensive information on the usage of Workspaces.

- **Views** (*CityModelAspect*) are used to display combined planning alternatives of different plannings. Only one planning alternative per planning can be selected.

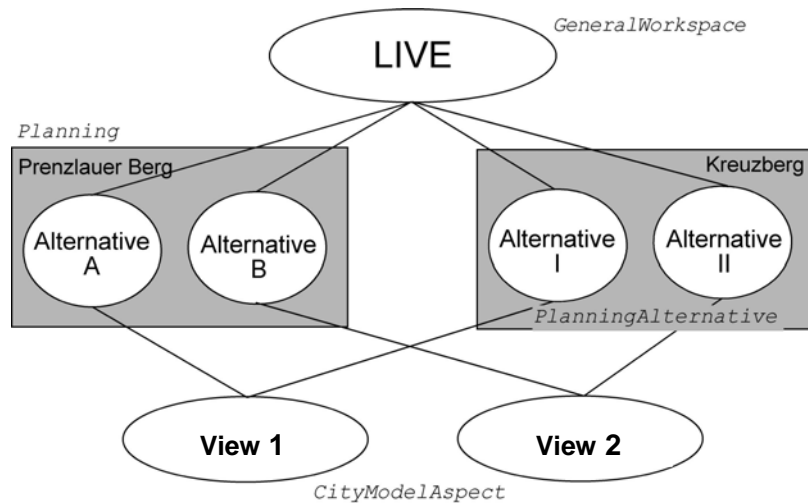


Figure 65: Overview of version and history management for parallel plannings and the corresponding terms

All objects that are displayed as an oval, in Figure 65, (*GeneralWorkspace*, *PlanningAlternative* and *CityModelAspect*) are realized as workspaces of the Oracle Workspace Manager and are managed in combination with additional metadata in separate tables. The UML chart in figure 65 shows the dependences and relations of the individual elements.

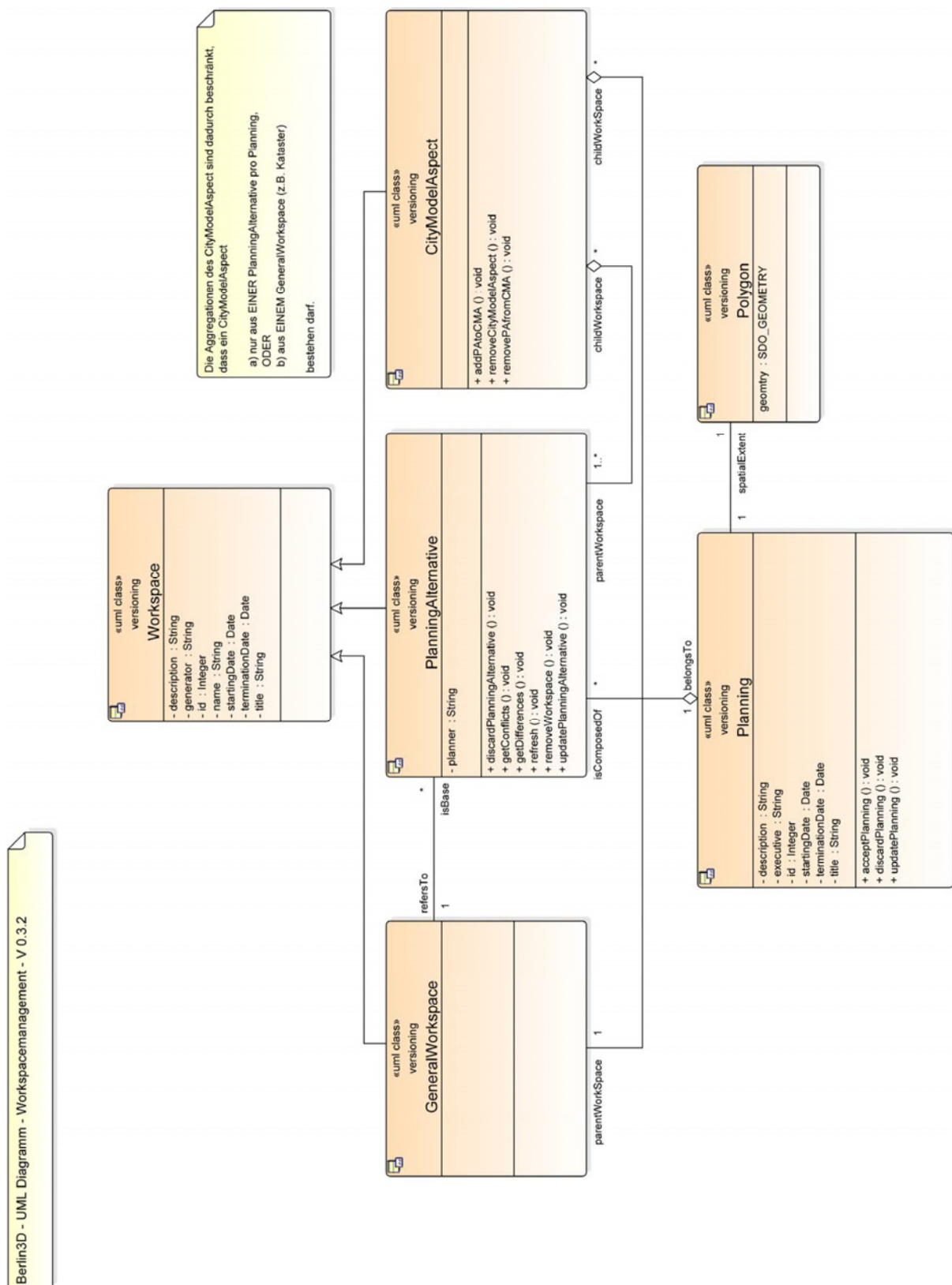


Figure 66: UML diagram comprising classes, attributes and methods for concurrent plannings

6.3 Realisation in Oracle

The described concept for parallel planning of different models is realized using functions of the Oracle Workspace Manager.

The database schema of the 3D geodatabase is extended by tables for the storage of the metadata of the version management (cf. Figure 67). The respective column names and column types can be seen from the graphical schema. Their meaning and function are explained in chapter 6.5 (administration program PlanningManager).

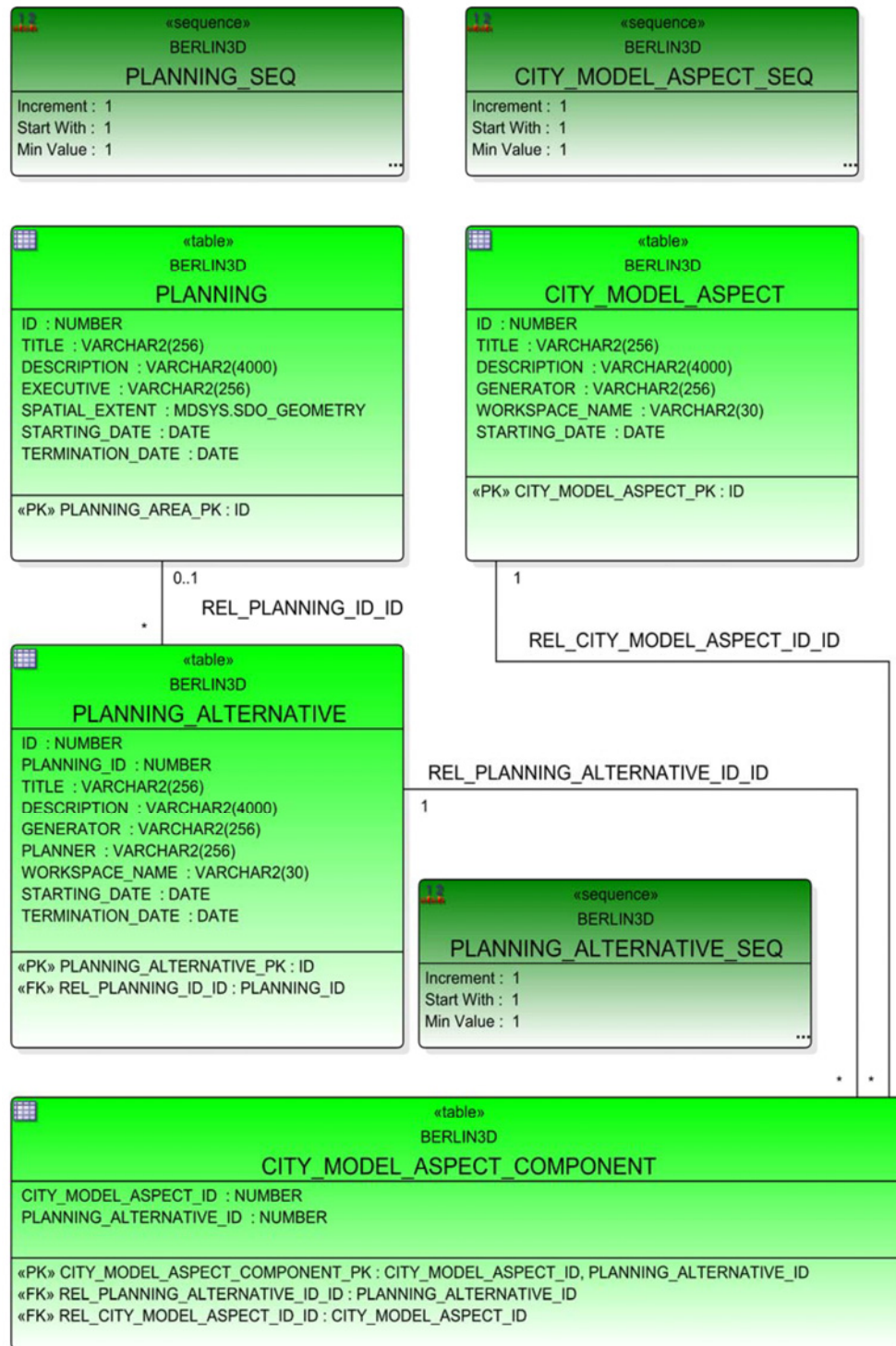


Figure 67: Database schema for parallel plannings

Because the actual versioning and history management is adopted from the Oracle Workspace Manager, the metadata tables and the corresponding procedures are used for a "versioning of the versioning". To ensure a complete version and history documentation, at no time data is deleted. That means that concluded plannings and planning alternatives are marked "finished" but the associated workspaces are not deleted. In case a view time in the past is selected in a database query, the data set that was valid at that time can be recovered.

To guarantee the integrity of the data, a direct writing access to the planning manager tables by users or application programs is not planned. Therefore, a number of procedures are provided for this purpose. Using these procedures it is possible for application programs to perform the version management within their program functionality. This way, a graphical user's interface (PlanningManager, cf. to chapter 6.5) was developed which also utilizes these methods.

Note: For the use of the procedures with SQL*Plus (command line) the parameter *SERVEROUTPUT* must be set to *ON* (command: *SET SERVEROUTPUT ON*), so that a return value can be displayed.

In the following chapters the procedures are introduced and explained. The following table gives an overview of the existing procedures.

Name of the function	Description	Chap.
AcceptPlanning	Adopt a planning alternative	6.3.4
AddPlanning	Starting a new planning	6.3.1
DiscardPlanning	Termination of a planning	6.3.3
UpdatePlanning	Updating of the metadata of a planning	6.3.2
AddPlanningAlternative	Starting a new planning alternative	6.3.5
DeleteAllPlanningAlternatives	Deletion of all planning altern. and workspaces	6.3.13
DeleteTermPlanningAlternatives	Deletion of all terminated planning alternatives and workspaces	6.3.14
DiscardPlanningAlternative	Termination of a planning alternative	6.3.7
GetAllConflicts	Query of the conflicts of all planning alternatives	6.3.11
GetAllDiff	Query of the differences of all planning alternatives	6.3.9
GetConflicts	Query of the conflicts of one planning alternative	6.3.10
GetDiff	Query of the differences of a planning alternative	6.3.8
RefreshPlanningAlternative	Updating of the data of one planning alternative.	6.3.12
UpdatePlanningAlternative	Updating of the metadata of a planning alternative.	6.3.6
AddCityModelAspect	Opening of a CityModelAspect (CMA)	6.3.15
AddPAtoCMA	Adding a planning alternative to a CMA	6.3.17
DeleteAllCityModelAspects	Deletion of all CityModelAspects including their workspaces	6.3.19
DeleteCityModelAspect	Deletion of one CityModelAspects including its workspace	6.3.16
RemovePAfromCMA	Removing a planning alternative from a CMA	6.3.18

Table 15: Functions for Planning Manager

6.3.1 AddPlanning

This procedure generates a planning, by creating a data record in the table PLANNING. If no spatial boundary is to be saved for the planning, the parameter must be given the value NULL.

Syntax:

```
AddPlanning (
    title           IN VARCHAR2,
    description      IN VARCHAR2,
    executive        IN VARCHAR2,
    spatialExtent    IN MDSYS.SDO_GEOMETRY);
```

Parameters:

title	Short name of the planning which is indicated in the menus of application programs (e.g. PlanningManager). It should be short and unique.
description	Description of the planning (max. 4000 characters)
executive	Responsible person / authority for initiation of the planning
spatialExtent	Spatial boundary of the planning area

6.3.2 UpdatePlanning

The procedure changes the parameters title, description, responsible authority and spatial boundary of an existing planning. All previously existing contents are overwritten.

Syntax:

```
UpdatePlanning (
    id              IN NUMBER
    title           IN VARCHAR2,
    description      IN VARCHAR2,
    executive        IN VARCHAR2,
    spatialExtent    IN MDSYS.SDO_GEOMETRY);
```

Parameters:

id	ID of the planning whose metadata is to be changed
title	Short name of the planning
description	Description of the planning (max. 4000 characters)
executive	Responsible person / authority for initiation of the planning
spatialExtent	Spatial boundary of the planning area

6.3.3 DiscardPlanning

The procedure terminates a planning, by setting a termination date for all alternatives and for the planning itself. For the workspaces of the planning alternatives savepoints with the name "terminated" are set. The workspaces are **not** deleted! The procedure is only executed if the planning is still active. Only active alternatives are terminated.

Syntax:

```
DiscardPlanning (id IN NUMBER);
```

Parameters:

id	ID of the planning to be terminated
----	-------------------------------------

6.3.4 AcceptPlanning

This procedure terminates a planning. The selected planning alternative is transferred to the parent workspace LIVE and therewith into the basic data set. Afterwards a termination date is set for all alternatives and the planning itself. For the workspaces of the alternatives, savepoints with the name "terminated" are set. The workspaces are **not** deleted!

The procedure is only executed if the specified planning is still active.

Syntax:

```
AcceptPlanning (
    planningId           IN NUMBER,
    planningAlternativeId IN NUMBER);
```

Parameters:

planningId	ID of the planning to be terminated
planningAlternativeId	ID of the planning alternative to be accepted

6.3.5 AddPlanningAlternative

The procedure opens a data record in the table PLANNING_ALTERNATIVE and creates a workspace. The workspace name consists of the label 'PA', the ID of the planning and the ID of the planning alternative (e.g. PA_37_5) and is derived from the workspace LIVE.

The procedure is only executed if the specified planning is still active.

Syntax:

```
AddPlanningAlternative (
    planningId   IN NUMBER,
    title        IN VARCHAR2,
    description   IN VARCHAR2,
    generator     IN VARCHAR2,
    planner      IN VARCHAR2);
```

Parameters:

planningId	ID of the planning that the new alternative is to be added to
title	Short name of the planning which is indicated in the menus of application programs (e.g. PlanningManager). It should be short and unique.
description	Description of the alternative to be added (max. 4000 characters)
generator	Name of the person, who created the alternative in the database
planner	Name of the author of this alternative (planner, architect)

6.3.6 UpdatePlanningAlternative

This procedure changes the parameters title, description, database generator and planner of an existing planning alternative. All existing entries are overwritten.

Syntax:

```
UpdatePlanningAlternative (
    id           IN NUMBER,
    title        IN VARCHAR2,
    description   IN VARCHAR2,
    generator     IN VARCHAR2,
    planner      IN VARCHAR2);
```


Parameters:

<code>id</code>	ID of the planning alternative which should be changed.
<code>title</code>	New short name of the planning alternative
<code>description</code>	New description of the planning alternative (max. 4000 characters)
<code>generator</code>	New name of the person, that created the planning alternative in the database
<code>planner</code>	New name of the author of this planning alternative (planner, architect)

6.3.7 DiscardPlanningAlternative

The procedure terminates a planning alternative. A termination date is set in the table `PLANNING_ALTERNATIVE` but the corresponding workspace is not deleted. Instead a savepoint with the name "terminated" is set. The procedure is only executed if the planning has not been terminated yet.

Syntax:

```
DiscardPlanningAlternative (id IN NUMBER);
```

Parameters:

<code>id</code>	ID of the planning alternative to be terminated
-----------------	-------------------------------------------------

6.3.8 GetDiff

The procedure returns the total number of tuples in the specified workspace that were changed compared to the basic data set (workspace LIVE). The value is calculated by adding up the differences in all versionized tables (e.g., `BUILDINGS`, `CITYOBJECT` etc.).

On the one hand, this number is an indicator for the amount of updates carried out in a planning alternative of the 3D city model. On the other hand, it is a measure for the complexity of the transfer of a planning alternative into the LIVE workspace.

Syntax:

```
GetDiff (workspace IN VARCHAR2);
```

Parameters:

<code>workspace</code>	Name of the workspace (e.g. <code>PA_21_3</code>) whose differences to LIVE are to be counted.
------------------------	-------------------------------------------------------------------------------------------------

6.3.9 GetAllDiff

This procedure calls the procedure `GetDiff` for all workspaces which are associated to planning alternatives. Thus, the result gives an impression of the processing status and the processing volume of all plannings of the entire database.

Syntax:

```
GetAllDiff ();
```

Parameters:

None

6.3.10 GetConflicts

The procedure returns the total number of tuples that were changed in the workspace LIVE as well as in the specified workspace. The value is calculated by adding up the conflicts in all versionized tables (e.g. `BUILDINGS`, `CITYOBJECT` etc.).

That means, the function indicates whether a transfer of the planning alternative into the LIVE workspace (Merge) or an updating of the data set of a planning alternative with the LIVE workspace (Refresh) is possible. Merge and Refresh can only be carried out for workspaces between which there are no conflicts.

Syntax:

```
GetConflicts (workspace IN VARCHAR2);
```

Parameters:

workspace Name of the workspace (e.g. PA_21_3)

6.3.11 **GetAllConflicts**

The procedure calls the procedure GetConflicts for all workspaces that are associated to planning alternatives. The result gives an impression of the possibility to conclude the plannings and to transfer them to the basic data set (workspace LIVE).

Syntax:

```
GetAllConflicts ();
```

Parameters:

None

6.3.12 **RefreshPlanningAlternative**

This procedure updates the data of the workspace of the specified planning alternative with those of the LIVE-workspace. This is necessary if the data in LIVE has changed since generation of the planning alternative. This is for example the case when another planning alternative was imported into the workspace LIVE. Changes in the workspace LIVE are not automatically adopted in the child-workspaces (planning alternatives)!

The call of this procedure is only possible if no conflicts exist between the LIVE workspace and the workspace of the specified planning alternative. Only the tuples in the versioned tables which are younger (newer) in LIVE than in the workspace of the planning alternative are changed. The number of these tuples can be analyzed beforehand using the procedure GetDiff.

Prior to the update of the workspace, a savepoint with the name "refreshed" is set (or overwritten). This allows saving the date of the last call of the procedure.

Syntax:

```
RefreshPlanningAlternative (id IN NUMBER);
```

Parameters:

id ID of the planning alternative

6.3.13 **DeleteAllPlanningAlternatives**

The procedure deletes all the workspaces recorded in the table PLANNING_ALTERNATIVE and the corresponding tuples in the table. This way, all planning alternatives and their workspaces are deleted!

Attention: This procedure deletes all data of the workspaces irrevocably and serves for optimization of the system performance and deletion of the database contents only.

Syntax:

```
DeleteAllPlanningAlternatives();
```

Parameters: None

6.3.14 DeleteTermPlanningAlternatives

This procedure deletes all terminated workspaces recorded in the table `PLANNING_ALTERNATIVE` and the corresponding tuples in the table. That means, all terminated planning alternatives and their workspaces are deleted!

Attention: This procedure deletes all data of the affected workspaces irrevocably and serves for optimization of the system performance and thinning / optimization of the data!

Syntax:

```
DeleteTermPlanningAlternatives();
```

Parameters:

None

6.3.15 AddCityModelAspect

The procedure generates a new `CityModelAspect`. It writes a new tuple into the corresponding table and generates a new workspace. The name of the workspaces consists of the label 'CMA' and the ID of the `CityModelAspect` (`CMA_CMAID`).

`CityModelAspects` are used only for simultaneous consideration of multiple planning alternatives and are therefore of temporary nature.

Syntax:

```
AddCityModelAspect (
    title                IN VARCHAR2,
    description           IN VARCHAR2,
    generator             IN VARCHAR2,
    planningAlternativeId IN NUMBER);
```

Parameters:

<code>title</code>	Short name of the view
<code>description</code>	Description of the view to be generated (max. 4000 characters)
<code>generator</code>	Name of the person, who generated the view
<code>planningAlternativeId</code>	ID of the planning alternative to be displayed

6.3.16 DeleteCityModelAspect

This procedure deletes a `CityModelAspect`. The tuples in the corresponding metadata tables (`CITY_MODEL_ASPECT` and `CITY_MODEL_ASPECT_COMPONENT`) are deleted as well as the associated workspace. However, by deleting `CityModelAspects` neither LIVE data nor planning data is deleted.

Syntax:

```
DeleteCityModelAspect (id IN NUMBER);
```

Parameters:

<code>id</code>	ID of the <code>CityModelAspect</code> to be deleted
-----------------	------------------------------------------------------

6.3.17 AddPAtoCMA

The procedure adds a planning alternative to a `CityModelAspect`. The procedure is only executed if the specified planning alternative and the corresponding planning are still active. In case an alternative of the same planning is already in use, the procedure is terminated.

Syntax:

```
AddPAtoCMA(
    cityModelAspectId      IN NUMBER,
    planningAlternativeId  IN NUMBER);
```

Parameters:

cityModelAspectId	ID of the CityModelAspect that the planning alternative is to be associated to.
planningAlternativeId	ID of the planning alternative

6.3.18 RemovePAfromCMA

The procedure deletes a planning alternative from a CityModelAspect.

Syntax:

```
RemovePAfromCMA(
    cityModelAspectId      IN NUMBER,
    planningAlternativeId  IN NUMBER);
```

Parameters:

cityModelAspectId	ID of the CityModelAspect from which the planning alternative is to be deleted.
planningAlternativeId	ID of the alternative to be deleted

6.3.19 DeleteAllCityModelAspects

The procedure deletes all CityModelAspects. The tuples in the corresponding metadata tables (CITY_MODEL_ASPECT and CITY_MODEL_ASPECT_COMPONENT) are deleted as well as the associated workspaces. Because CityModelAspects are conceived only as temporarily defined views, no LIVE or planning data is deleted.

Syntax:

```
RemoveAllCMAWorkspaces();
```

Parameters:

None

6.4 Administration program "PlanningManager"

For a more convenient management of plannings, planning alternatives and CityModelAspects the graphic user interface *PlanningManager* can be used. The program requires a Java installation, version 1.4 or newer, and can be launched without installation by double click on the file *PlanningManager.exe*. Alternatively the program can be launched via the command line. This way the database parameters can be passed directly enabling the program to create a database connection immediately at the start of the program.

```
PlanningManager.exe <host> <port> <sid> <user> <password>
```

or

```
java -jar PlanningManager.jar <host> <port> <sid> <user> <password>
```

Application programs that work with the 3D geodatabase can access the functionality of the PlanningManager via the command line call. In particular, this is necessary when working in or with different planning alternatives. Further details can be found in chapter 3.6 „Use in application programs“.

If the program is called without parameters, the database connection can be created via the menu item "database" – "connect". The connection state is indicated in the left lower corner of the program window ("connected" or "disconnected").

While creating the connection to the database the existing and not yet terminated plannings and planning alternatives are read from the corresponding tables and displayed in a treelike view. By choosing a planning or a planning alternative, the corresponding metadata can be displayed.

6.4.1 Plannings

The PlanningManager allows managing all plannings that are not terminated. Figure 68 shows the metadata view of a planning. The displayed information has the following meaning:

- ID ID of the planning (automatic sequential numbering)
- Titel Name of the planning (256 characters)
- Beschreibung Description of the planning (4000 characters)
- Ausführende Stelle Authority / supervision of the planning (256 characters)
- Begrenzung Spatial boundary polygon (2D) of the planning (coordinate list see below)
- Beginn Date and time of the creation of the planning
- Ende Date and time of the termination of a planning

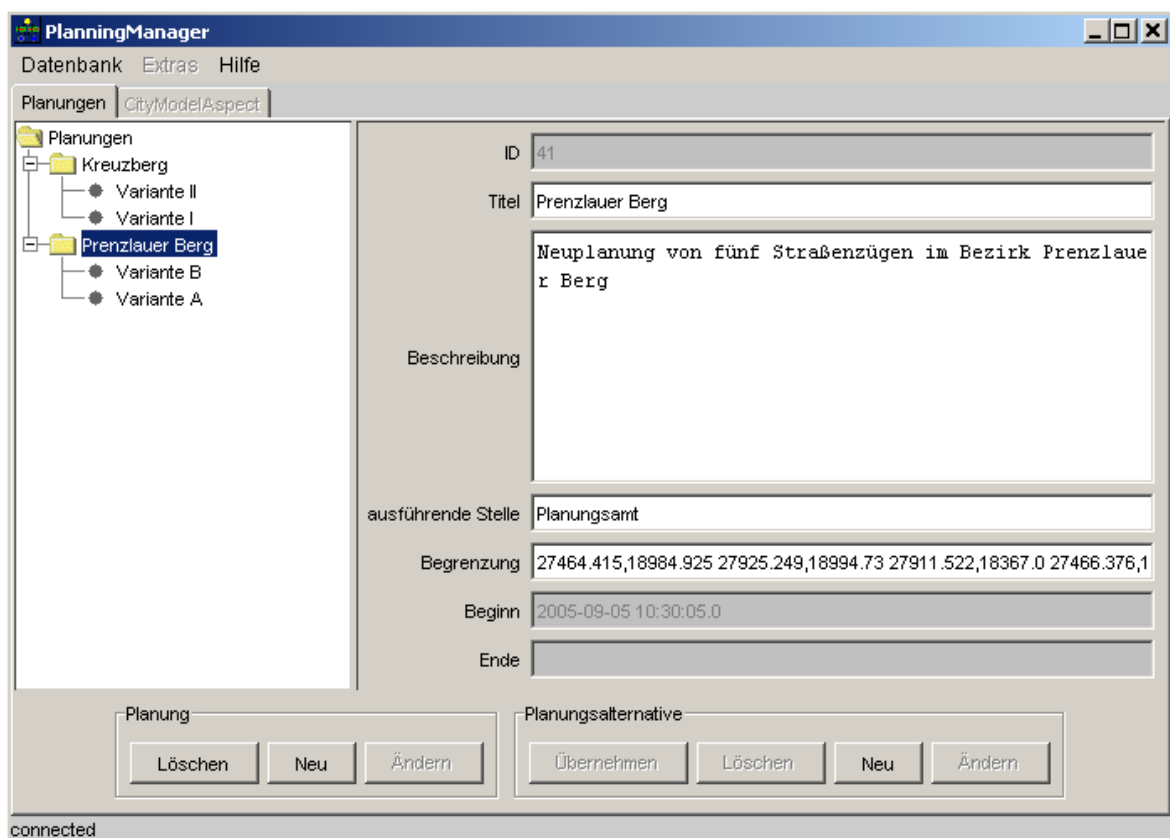


Figure 68: PlanningManager: Interface for the management of plannings

The information in the fields with grey background is generated automatically and cannot be edited by the user.

The spatial boundary of a planning area is given by a list of coordinates. These describe a polygon by which the planning area is delimited. Following separators are to be used:

- The decimal separator is a dot (".")
- Easting and northing are to be separated by a comma (",")
- Coordinate pairs (Points) are to be separated by blanks (" ")

The corners of the polygon are to be described by easting and northing, an elevation is not allowed. The entered polygon is saved in the table PLANNING in the corresponding field of the type MDSYS.SDO_GEOMETRY.

In contrast to Oracle Spatial when entering the polygon points the first coordinate pair has not to be repeated at the end of the list.

For the management of the metadata of plannings the following buttons are available. They are activated or deactivated depending on the context:

- **Löschen** Deletion of the selected planning. A planning is deleted, by writing the current date in the table PLANNING into the column `TERMINATION_DATE` of the corresponding tuple. The tuple continues to exist but the planning is not displayed in the PlanningManager anymore.
- **Neu** Definition of a new planning. A new planning is generated, by writing the entered and the automatically generated metadata under a new ID into the table PLANNING.

Note: If no spatial limitation is to be associated to the new planning, *"null"* (without quotation marks) is to be entered into the corresponding field or the field is to be left empty!
- **Ändern** Altered entries are adopted. The changed metadata is written into the corresponding row of the table PLANNING.

6.4.2 Planning alternatives

The PlanningManager allows managing all planning alternatives that are not terminated. Figure 3.5 shows the metadata view of a planning alternative. The displayed information has the following meaning:

- **ID** ID of the planning alternative (automatic sequential numbering)
- **Titel** Name of the planning alternative (*256 characters*)
- **Beschreibung** Description to the planning alternative (*4000 characters*)
- **In DB eingefügt von** Name / username of the person who created the planning alternative in the database (*256 characters*)
- **Planer** Name of the planner / architect of the planning alternative (*256 characters*)
- **Workspacename** Name of the workspace of the planning alternative. The name is generated automatically and consists of the label `PA_`, the ID of the planning, which the planning alternative is associated to, and the ID of the planning alternative. (That means, `PA_25_21` is the workspace associated to the planning alternatives 21, which in turn is associated to the planning 25.)

By pressing the button, the name of the workspaces is copied to the clipboard. That means, it can easily be transferred to other application programs.

If geodata of a planning alternative are to be changed (in SQL*Plus or an application program), first of all the corresponding workspace must be selected. This is done via the command:

```
EXEC DBMS_WM.GoToWorkspace( '<Workspacename'> ).
```

- Beginn
- Ende

Date and time of the creation of the planning alternative

Date and time of the termination of the planning alternative

Figure 69: PlanningManager: Interface for the management of plannings

The information in the fields with grey background is generated automatically and cannot be edited by the user.

For the management of the metadata of planning alternatives the following buttons are available. They are activated or deactivated depending on the context:

- **Übernehmen** Accept the selected planning alternative. The content of the corresponding workspace is transferred into the workspace LIVE. **The data stock of LIVE (basic data set) is thus changed!** Furthermore, setting the termination date terminates the planning alternative, all competing planning alternatives and the planning itself. The respective tuples remain in the tables PLANNING and PLANNING_ALTERNATIVE. However, they are not displayed in the PlanningManager anymore.
- **Löschen** Deletion of the selected planning alternative. A planning alternative is deleted, by writing the current date in the table PLANNING_ALTERNATIVE into the column TERMINATION_DATE of the corresponding tuple. Thus, the tuple continues to exist. A savepoint with the name "terminated" is set for the corresponding workspace. The workspace is not deleted, but the planning alternative is not displayed in the PlanningManager anymore.
- **Neu** Definition of a new planning alternative. A new planning alternative can only be generated if a planning is selected. This is because an alternative is assigned to exactly one planning. A planning alternative is generated, by writing the entered and the automatically generated metadata under a new ID into the table PLANNING_ALTERNATIVE. A new workspace is created in the database.
- **Ändern** Altered entries are adopted. The changed metadata is written into the corresponding row of the table PLANNING_ALTERNATIVE.

6.4.3 Interface for the management of planning alternatives

Some functionality for planning alternatives is offered under the menu item *Extras*. When choosing a planning, the menu item is inactive.

- Differenzen Query of the number of differences between the workspace LIVE and the workspace of the selected planning alternative (cf. 6.3.8 procedure GetDiff). The call of the function can take up some time (several minutes)!
- Konflikte Query of the number of conflicts between the workspace LIVE and the workspace of the selected planning alternative (cf. 6.3.10 procedure GetConflicts). The call of the function can take up some time (several minutes)!
- Aktualisierungsdatum Date of the last update of the workspaces of the selected planning alternative (cf. RefreshPlanningAlternative 6.3.12). If no update has been carried out yet, the date of the creation of the planning alternative is displayed.
- Aktualisieren Updating of the workspace of the planning alternative with the current data of the workspace LIVE.

6.5 Conflict management

On the database side planning alternatives are realized as workspaces of the Oracle Workspace Manager. A workspace is the logical aggregation of the updates of the rows of versioned tables (cf. "Oracle - Database Application Developer's Guide - Workspace Manager"). All workspaces for planning alternatives are derived from the workspace LIVE. That means that there are multiple parallel versions of the original dataset LIVE corresponding to the different planning alternatives. The dataset is modified at the level of the respective planning alternatives.

6.5.1 Differences

The individual updates of the records of a workspace lead to differences between the planning alternatives (workspaces), as well as between the planning alternatives and the original data set from which they are derived. To get an overview of how many records must be changed in case of an adoption of a planning alternative into the original dataset LIVE (Merging), the menu item *Differences* in the menu *Extras* of the PlanningManager can be selected. This tool analyzes the differences between the workspace of the currently selected planning alternative and the workspace LIVE.

6.5.2 Conflicts

If a data record is changed in the original dataset of the LIVE workspaces as well as in the workspace of a planning alternative, a conflict arises. The transfer of the planning alternative into the original dataset is not possible as long as conflicts exist. The menu item *Conflicts* in the menu *Extras* of the PlanningManager allows to count conflicts just like differences.

Note: Conflicts can appear if those tuples of the original dataset are changed - after a planning alternative has been created - which are also changed in that planning alternative! Via the spatial boundary of the planning these cases can generally be avoided. Conflicts concern the object level and must be solved by all means prior to the update of a planning alternative or the adoption of a planning alternative. Naturally, conflicts are solved individually. This

should happen in an application program with graphic representation of both objects. The PlanningManager offers no functionality for the solving of conflicts.

6.6 Use in application programs

In order to be able to access a planning alternative, an application program must switch to the workspace of the planning alternative prior to the query of the database. This is done via the SQL statement

```
EXEC DBMS_WM.GoToWorkspace ( '<name>' ),
```

wherein <name> is to be replaced with the name of the workspaces (e.g., *PA_32_25*). The name of the workspace of a planning alternative can be displayed and copied to the clipboard through the administration program PlanningManager. For this purpose the call of the PlanningManager from the application program with handing over of the database parameters is particularly useful.

7 Tools for raster data import and export

The import and export program of version 1 of the 3D citydb provides a number of functions for reading and writing digital terrain models and aerial images. DTMs and aerial images are processed as georeferenced raster files (TIFF format with additional Worldfile).

Please note, that the raster importer/exporter has been adopted from the previous release of the 3D citydb. This chapter is an English translation of the documentation provided in [Plümer et al. 2005]. It also refers to some functionalities (especially shapfile processing) which are untested with version 2 of the 3D citydb and may not work properly. The tool can be downloaded from 3D City Database homepage (<http://www.3dcitydb.org> – scripts & tools).

The import functionality is used for adding new records to the 3D geodatabase. Imported geodata are added to the database as new objects, receiving new, unique object-IDs. Because no records in the database are replaced, the import tool is only of limited use for the continual updating of the data. The database is filled with raster data by importing single raster data tiles, which are eventually merged to form one big, homogeneous raster data object (aerial image or DTM).

Using the export functionality of the program excerpts of the 3D geodatabase can be generated. The spatial region to be exported can generally be selected by indicating the surrounding rectangle (Bounding box). In addition, when it comes to outputting specific buildings, object-IDs or the affiliation to a CityObject-group can also be used as a selection criterion.

For all import and export processes, the Oracle-Workspace from which the data is to be taken or into which the data is to be imported can be explicitly stated (cf. chapter 6). This way it is possible for example to systematically add new buildings to a planning alternative.

7.1 System requirements

The import/export tool requires Java Runtime Environment Standard Edition (SE) version 1.4.2 (JRE 1.4.2) or higher. JRE 1.4.2 can be downloaded via the following link:

<http://java.sun.com/j2se/1.4.2/download.html>

and must be installed prior to the execution of the import/export tool. Please, make sure that you download the right JRE according to your operating system (e.g. Windows, Linux, Solaris). JRE 1.4.2 requires approximately 15 MB of hard disk space.

The import/export tool needs approx. 9 MB of hard disk space and requires at least 256 MB of working memory (RAM). This is sufficient, for importing and exporting small Shapefiles (recommended size: smaller than 10 MB) or raster data (recommended size: smaller than 50 MB). For the import of big images, for example the supplied images in original size, a minimum of 1 GB of working memory is recommended.

Please note that additional hard disk space is required for the export, depending on the size of the dataset to be exported. As a guide value 500 MB should be provided for exported data. After completion of the export process, these data can be relocated. Furthermore temporary hard disk space is required for the tiling of images, depending on the size of the image to be tiled. (When tiling an image the size of the temporary hard disk space should be at least as big as the original image.)

7.2 User interface

The application is launched with the following statement:

```
java.exe -Xms256m -Xmx1024m -jar berlin3d_tool.jar
         jdbc:oracle:thin:@<myserver>:1521:<myDBInstance>
```

Hint: Users of a Windows operating system can also use the programs "javaw.exe" or "start java.exe". This way the command window for the start of the application will be closed.

The execution command comprises four parameters: The parameters of the Java Virtual machine (JVM) for the main memory (256 MB as a start value and 1 GB as the maximum working memory), the jar-parameter of the application and the database parameters. In the latter the parameter <myserver> is to be replaced with the IP address of the database server and the parameter <myDBInstance> is to be replaced with the SID of the database. If necessary, these values can be customized by the system administrator.

All required system libraries are stored in the subdirectory ./libs/. They are loaded automatically. After a successful start the graphical user interface appears (cf. Figure 70).

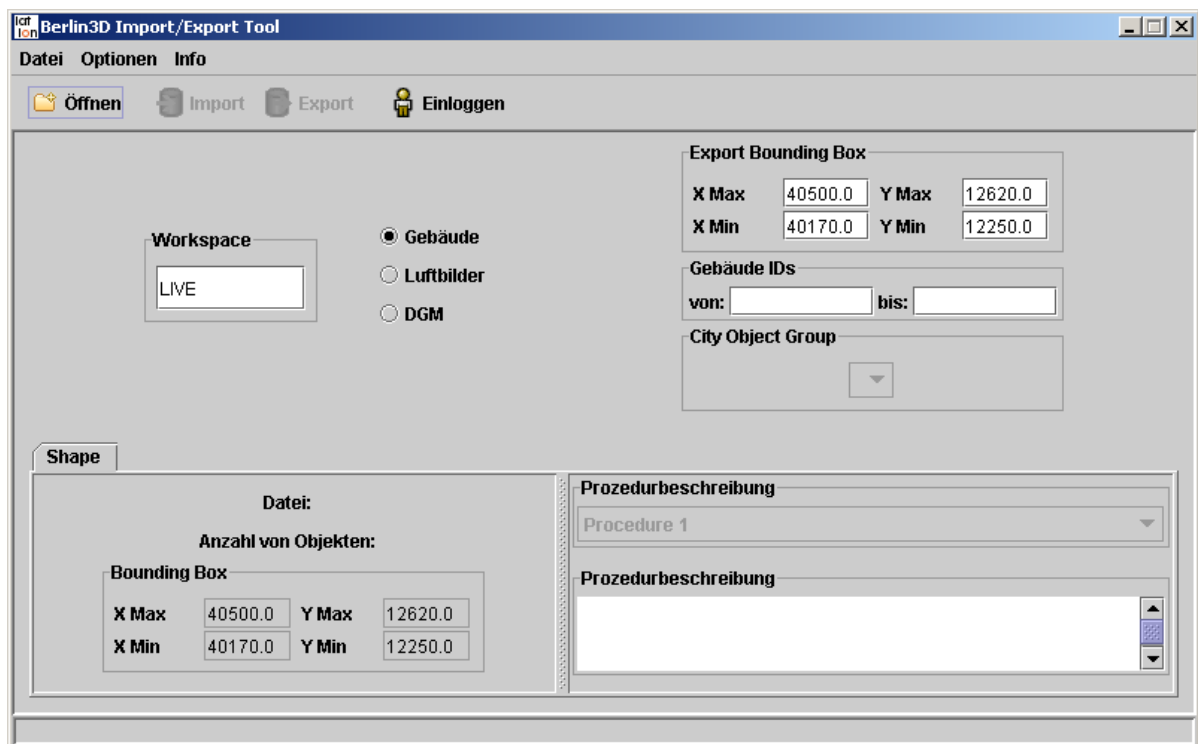


Figure 70: The raster import / export tool

7.2.1 The login dialog

In order to import or export data the user must first logon to the database. This is done by pressing the button *Einloggen*, entering login name and password and confirming the entry with *OK* (cf. Figure 71).

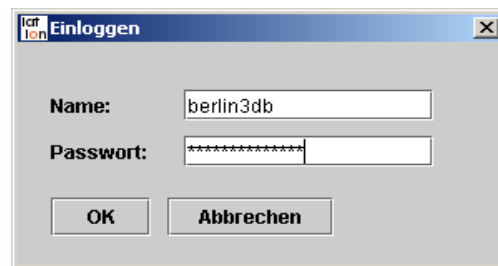


Figure 71: Login mask

Subsequently the application establishes a connection to the database. As soon as the connection is established, the list of import conversion procedures (title "Prozedurbeschreibung" or "Prozedurname") and the select box "CityObjectGroup" are activated. The procedure-select-list is used for the import of Shapefiles. As this functionality is deprecated it will not be explained in more detail. The German version can still be read in the original documentation [Plümer et al., 2005].

7.3 Import

7.3.1 Import of raster data

The storage concept of the 3D-Geo-DB is designed to manage aerial images as well as DTMs for every level of detail as a homogeneous raster data object in the database. To generate such a big raster, the raster data must be available in the form of single "raster data tiles" which must first be imported individually into the database using the import/export program. After all tiles for one level of detail have been transferred to the database, the user can initiate the generation of the overall raster by calling specific PL/SQL procedures. The call of these procedures is explained in the following paragraphs 7.3.1.1 and 7.3.1.2.

In order to import the raster data tiles, the user must be logged on to the system. The data can either be imported as aerial images or as digital terrain models (DTMs). In both cases, a so-called 'Worldfile' with coordinate information is required. This file must have the same name as the image file but ".tfw" instead of ".tif" (for TIFF) as file extension.

The tool assumes that the separator for decimal places in the Worldfile is a dot "." and not a comma! Commas are automatically converted into dots during the read-in process.

After selection of a TIFF image the user clicks on import to start the reading process. This can take up some time, for example approx. 15 minutes for big files (500 MB) depending on the server capacity and network connection. Please note, that images must be loaded individually.

In case an error message appears during read-in of very big raster data files concerning a lack of storage space, the raster data file can be divided into smaller tiles prior to the import (approximate value: from about 400 MB). For this purpose the tool offers a tiling function that can be found under the menu item *Options-> Tile image*. After clicking onto the respective menu button, the user is prompted to specify the image that is to be tiled. After entering the new name of the image, e.g. "new_Image", the original image is divided into four sub-images, which are named "new_Image_0_0.tif", "new_Image_0_1.tif", "new_Image_1_0.tif" and "new_Image_1_1.tif". The corresponding Worldfiles are generated automatically.

7.3.1.1 Mosaicking of DTMs

The generation of an overall raster object from single raster tiles is carried out completely within the Oracle-DBMS by means of Stored Procedures. Due to the large data volume that has to be relocated this process can last several hours, sometimes up to one day.

The realization of the so-called "mosaicking" is not controlled via the import/export program, but must be initiated by entering the respective command in an SQL input console, as for example the attached Oracle SQL*Plus or the Enterprise Manager console. First the user must log on to the 3D geodatabase via the console program. Afterwards he can call the procedure "mosaicRasterReliefInitial" which merges all imported DTM raster tiles – like in a mosaic – forming the overall raster object. In order to generate a homogeneous overall raster dataset a number of conditions must be satisfied: The tiles have to cover a rectangular area without gaps or overlaps. Furthermore the spatial resolution of all tiles must be identical.

In addition to the actual overall grid, the mosaicking procedure generates a RasterReliefObject that contains the former plus a relief object representing the overall DTM for a given level of detail (cf. modelling in paragraph 2.2.1.3.4 and DB schema in paragraph 2.3.2.6). Both geobjects are CityObjects, which are represented by an individual tuple in the table CITYOBJECT as well as an affiliated tuple in RASTER_RELIEF and RELIEF. The ID values of both CITYOBJECT tuples are automatically generated and, after successful execution of the procedure, displayed in the console window. They should be written down in a database documentation, because they are needed for later accesses (e.g. for export).

The syntax for calling the mosaic procedure is as follows:

```
execute mosaicRasterReliefInitial (<name>, <type>, <origin>, <LOD>);
```

The parameter <name> is a text string and denotes the name of the overall raster (e.g. 'Overall - DTM Berlin'). <Type> is also a text string and serves for a brief description of the type of the raster data as for example 'regular grid 0.5 m'. The text string <origin> contains information on the data source or the data supplier (e.g. 'photogrammetric image evaluation of HRSC data, RSS inc'). The parameter <LOD> may be an integer between 1 and 3, it specifies to which level of detail the DTM is to be assigned.

If all DTM tiles have been successfully loaded into the database with the import/export program, the LOD2 Overall DTM of Berlin can be generated, by entering the following commands in SQL*Plus (please enter the 2. command without line break):

```
set serveroutput on

execute mosaicRasterReliefInitial ('DTM of Berlin', 'Regular grid
0.5 m resolution', 'Photogrammetric evaluation, RSS Inc', 2);
```

7.3.1.2 Mosaicking of aerial images

The creation of the overall aerial image from single image tiles works analogously to the mosaicking of DTMs (see previous paragraph). Again, the user has to call a stored procedure of the 3D-Geo-DB. The mosaic procedure generates an Orthophoto object which contains the overall raster. The Orthophoto object is a CityObject which is represented by a tuple in the table CITYOBJECT as well as an affiliated tuple in the table ORTHOPHOTO. The ID is identical for both tuples. It is automatically generated and, after successful execution of the procedure, displayed in the console window. It should be written down in a database documentation, because it is needed for later accesses (e.g. for export). The syntax of the mosaicking procedure is as follows:

```
execute mosaicOrthophotosInitial (<name>, <type>, <origin>, <LOD>);
```

The parameter <name> is a text string and denotes the name of the overall raster (e.g. 'aerial image of Berlin'). <Type> is also a text string and serves for a brief description of the type of the raster data as for example 'True Orthophoto'. The text string <origin> contains information on the data source or the data supplier (e.g. 'HRSC-camera flight, RSS inc'). The parameter <LOD> may be an integer between 1 and 3, it specifies to which level of detail the aerial image is to be assigned.

If all image tiles have been successfully loaded into the database with the import/export program, the LOD2 Overall orthophoto of Berlin can be generated, by entering the following commands in SQL*Plus (please enter the 2. command without line break):

```
set serveroutput on  
  
execute mosaicOrthophotosInitial ('aerial picture of Berlin', 'True  
Orthophoto', 'HRSC camera flight, RSS inc', 2);
```

7.4 Export

7.4.1 Export of raster data

The export of raster data works analogously to the export of Shapefiles, besides that the information on building-ID and CityObjectGroup are irrelevant. That means, only the specifications of the bounding box are considered for the selection of the area to be exported.

By clicking the appropriate button (*Luftbilder* or *DGM*) the user can choose whether he wants to export image or DTM data.

After pressing the button *Export* the user is asked for the destination file. The file extension ".tif" is automatically added by the tool. Beside the image file, a Worldfile with the same name as the image but with the file extension ".tfw" is generated which contains the information for georeferencing of the image.

8 References

- 3D City Database, Weblink (accessed April 2009) <http://www.3dcitydb.org/>
- 3D City Database v2 / 3D City Database Import/ Export Tool. Weblink (accessed April 2009) <http://www.igg.tu-berlin.de/software/>
- Berlin 3D homepage (accessed April 2009): Weblink <http://www.virtual-berlin.de>
- CityGML homepage. (accessed April 2009): Weblink <http://www.citygml.org>
- Döllner, Jürgen / Buchholz, Henrik / Brodersen, Florian / Glander, Tassilo / Jütterschenke, Sascha / Klimetschek, Alexander (2005): Smart Buildings – A Concept for Ad-Hoc Creation and Refinement of 3D Building Models. In: Proceedings of the 1st International Workshop of 3D City Models, Bonn, Germany, June 2005
- Döllner, Jürgen / Kolbe, Thomas H. / Liecke, Falko / Sgouros, Takis / Teichmann, Karin (2006): The Virtual 3D City Model of Berlin - Managing, Integrating, and Communicating Complex Urban Information. In: Proceedings of the 25th Urban Data Management Symposium UDMS 2006 in Aalborg, Denmark, May 15-17. 2006.
- Gröger, Gerhard / Kolbe, Thomas H. / Czerwinski, Angela / Nagel, Claus (2008): OpenGIS City Geography Markup Language (CityGML) Encoding Standard, Version 1.0.0, International OGC Standard. Open Geospatial Consortium, Doc. No. 08-007r1 2008.
- Gröger, Gerhard / Kolbe, Thomas H. / Czerwinski, Angela (2007): Candidate OpenGIS® CityGML Implementation Specification (City Geography Markup Language), Version 0.4.0 Open Geospatial Consortium, Doc. No. 07-062 2007.
- Gröger, Gerhard / Kolbe, Thomas H. / Schmittwilken, Jörg / Stroh, Viktor / Plümer, Lutz (2005): Integrating versions, history and levels-of-detail within a 3D geodatabase. In: Gröger, Gerhard / Kolbe, Thomas H. (Hg.): Proceedings of the 1st intern. ISPRS/EuroSDR/DGPF-Workshop on Next Generation 3D City Models. EuroSDR, Bonn 2005.
- Kolbe, Thomas H. (2009): Representing and Exchanging 3D City Models with CityGML. In: Lee, Jiyeong / Zlatanova, Sisi (Ed.): Proceedings of the 3rd International Workshop on 3D Geo-Information, Seoul, Korea. Lecture Notes in Geoinformation & Cartography, Springer Verlag, 2009.
- Nagel, Claus / Stadler, Alexandra (2008): Die Oracle-Schnittstelle des Berliner 3D-Stadtmodells. In: Clemen, Christian (Ed.): Entwicklerforum Geoinformationstechnik 2008. , Shaker Verlag, Aachen, S. 197-221.
- Plümer, Lutz, / Gröger, Gerhard / Kolbe, Thomas H. / Schmittwilken, Jörg / Stroh, Viktor / Poth, Andreas / Taddeo, Ugo (2005): 3D-Geodatenbank Berlin, Dokumentation V1.0 Institut für Kartographie und Geoinformation der Universität Bonn (IKG), lat/lon GmbH. See <http://www.3dcitydb.org> (accessed April 2009).
- Stadler, Alexandra / Nagel, Claus / König, Gerhard / Kolbe, Thomas H. (2009): Making interoperability persistent: A 3D geo database based on CityGML. In: Lee, Jiyeong / Zlatanova, Sisi (Ed.): Proceedings of the 3rd International Workshop on 3D Geo-Information, Seoul, Korea. Lecture Notes in Geoinformation & Cartography, Springer Verlag, 2009.
- Whiteside, Arliss /Ed.) (2007): Definition identifier URNs in OGC namespace of the OGC, Best practice paper, document number: 07-093r1 (accessed April 2009) http://portal.opengeospatial.org/files/?artifact_id=24045

9 Appendix A – SQL Scripts

The following pages contain the listing of all scripts available for handling the database. For the sake of clarity the filetree is printed:

```

C:\BERLIN3D\ORACLESQL
ADD_CONSTRAINTS.sql
BUILD_SIMPLE_INDEX.sql
CREATE_DB.sql
CREATE_DB2.sql
CREATE_PLANNINGMANAGER.sql
CREATE_SEQUENCES.sql
DATABASE_SRS.sql
DISABLEVERSIONING.sql
DROP_DB.sql
DROP_PLANNINGMANAGER.sql
DUMMY_IMPORT.sql
ENABLEVERSIONING.sql
GEODB_REPORT.sql
HINT_ON_MISSING_SRS.sql
IMPORT_PROCEDURES.sql
MOSAIC.sql
OBJECTCLASS_INSTANCES.sql
SPATIAL_INDEX.sql
TRIGGER.sql

+---APPEARANCE
    APPEARANCE.sql
    APPEARANCE_SEQ.sql
    APPEAR_TO_SURFACE_DATA.sql
    SURFACE_DATA.sql
    SURFACE_DATA_SEQ.sql
    TEXTUREPARAM.sql

+---BUILDING
    ADDRESS.sql
    ADDRESS_SEQ.sql
    ADDRESS_TO_BUILDING.sql
    BUILDING.sql
    BUILDING_FURNITURE.sql
    BUILDING_INSTALLATION.sql
    OPENING.sql
    OPENING_TO_THEM_SURFACE.sql
    ROOM.sql
    THEMATIC_SURFACE.sql

+---CITYOBJECT
    CITYMODEL.sql
    CITYMODEL_SEQ.sql
    CITYOBJECT.sql
    CITYOBJECT_GENERICATTRIB.sql
    CITYOBJECT_GENERICATT_SEQ.sql
    CITYOBJECT_MEMBER.sql
    CITYOBJECT_SEQ.sql
    CITY_FURNITURE.sql
    EXTERNAL_REFERENCE.sql
    EXTERNAL_REF_SEQ.sql
    GENERALIZATION.sql
    GENERIC_CITYOBJECT.sql
    IMPLICIT_GEOMETRY.sql
    IMPLICIT_GEOMETRY_SEQ.sql
    OBJECTCLASS.sql
    SURFACE_GEOMETRY.sql
    SURFACE_GEOMETRY_SEQ.sql

+---CITYOBJECTGROUP
    CITYOBJECTGROUP.sql
    GROUP_TO_CITYOBJECT.sql

+---DTM
    BREAKLINE_RELIEF.sql
    DTM_SEQ.sql
    MASSPOINT_RELIEF.sql
    RASTER_RELIEF.sql
    RASTER_RELIEF_IMP.sql
    RASTER_RELIEF_IMP_RDT.sql
    RASTER_RELIEF_RDT.sql
    RASTER_RELIEF_RDT_id_trigger.sql
    RASTER_RELIEF_RDT_IMP_id_trigger.sql
    RELIEF.sql
    RELIEF_COMPONENT.sql
    RELIEF_FEATURE.sql
    RELIEF_FEAT_TO_REL_COMP.sql
    TIN_RELIEF.sql

+---LANDUSE
    LAND_USE.sql
    PLANT_COVER.sql
    SOLITARY_VEGETAT_OBJECT.sql

+---ORTHOPHOTO
    CREATEDMLTRIGGER_ORTHOPHOTOPROPERTY.sql
    ORTHOPHOTO.sql
    ORTHOPHOTO_IMP.sql
    ORTHOPHOTO_RDT.sql
    ORTHOPHOTO_RDT_id_trigger.sql
    ORTHOPHOTO_RDT_IMP.sql
    ORTHOPHOTO_RDT_IMP_ID_TRIGGER.sql
    ORTHOPHOTO_SEQ.sql

+---PLANNINGMANAGER
    CREATE_CITYMODELASPECT_PROCEDUREBODYS.sql
    CREATE_CITYMODELASPECT_PROCEDURES.sql
    CREATE_CONSTRAINTS.sql
    CREATE_PLANNINGALTERNATIVE_PROCEDUREBODYS.sql
    CREATE_PLANNINGALTERNATIVE_PROCEDURES.sql
    CREATE_PLANNING_PROCEDUREBODYS.sql
    CREATE_PLANNING_PROCEDURES.sql
    CREATE_SPATIAL_INDEX.sql
    CREATE_TABLES.sql
    CREATE_UTIL_PROCEDURES.sql

+---SYSDBA
    CREATE_USER_BERLIN3D.sql
    SOLDNER_BERLIN_SRS.sql
    SOLDNER_BERLIN_SRS_10G_R2.sql

+---TRANSPORTATION
    TRAFFIC_AREA.sql
    TRANSPORTATION_COMPLEX.sql

+---WATERBODY
    WATERBODY.sql
    WATERBOD_TO_WATERBND_SRF.sql
    WATERBOUNDARY_SURFACE.sql
  
```

9.1 Database

CREATE_DB.sql

```

-- CREATE_DB.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
  
```

```

-- Copyright:   (c) 2007-2008, Institute for Geodesy and Geoinformation Science,
--              Technische Universität Berlin, Germany
--              http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description                                     | Author
-- 2.0.1   | 2008-06-28 | include query for versioning enable             | TKol
-- 2.0.0   | 2007-11-23 | release version                                 | TKol
--                                                | GKoe
--                                                | CNag
--                                                | ASa
SET SERVEROUTPUT ON
SET FEEDBACK ON

prompt
prompt
accept SRSNO NUMBER DEFAULT 81989002 PROMPT 'Please enter a valid SRID (Berlin: 81989002): '
prompt Please enter the corresponding SRSName to be used in GML exports
accept GMLSRNAME CHAR DEFAULT 'urn:ogc:def:crs,crs:EPSG:6.12:3068,crs:EPSG:6.12:5783' prompt
' (Berlin: urn:ogc:def:crs,crs:EPSG:6.12:3068,crs:EPSG:6.12:5783): '
accept VERSIONING CHAR DEFAULT 'no' PROMPT 'Shall versioning be enabled (yes/no, default is
no): '
prompt
prompt

VARIABLE SRID NUMBER;
VARIABLE CS_NAME VARCHAR2(256);
VARIABLE BATCHFILE VARCHAR2(30);

WHenever SQLERROR Continue;

BEGIN
    :BATCHFILE := 'HINT_ON_MISSING_SRS';
END;
/

BEGIN
    SELECT SRID,CS_NAME INTO :SRID,:CS_NAME FROM MDSYS.CS_SRS
    WHERE SRID=&SRSNO;

    IF (:SRID = &SRSNO) THEN
        :BATCHFILE := 'CREATE_DB2';
    END IF;
END;
/

-- Transfer the value from the bind variable to the substitution variable
column mc new_value BATCHFILE2 print
select :BATCHFILE mc from dual;

START &BATCHFILE2

```

CREATE_DB2.sql

```
-- CREATE_DB2.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stroh@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008, Institute for Geodesy and Geoinformation Science,
--               Technische Universität Berlin, Germany
--               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description                                     | Author
-- 2.0.1   | 2008-06-28 | versioning is enabled depending on var         | TKol
-- 2.0.0   | 2007-11-23 | release version                                | TKol
--                                                | GKoe
--                                                | CNag
--                                                | Asta
--
SET SERVEROUTPUT ON
SET FEEDBACK ON

VARIABLE VERSIONBATCHFILE VARCHAR2(30);

-- This script is called from CREATE_DB.sql and it
-- is required that the three substitution variables
-- &SRSNO, &GMLSRSNAME, and &VERSIONING are set properly.

--//CREATE TABLES
@@DATABASE_SRS.sql

INSERT INTO DATABASE_SRS(SRID,GML_SRS_NAME) VALUES (&SRSNO,'&GMLSRSNAME');
COMMIT;

@@CITYOBJECT/CITYMODEL.sql
@@CITYOBJECT/CITYOBJECT.sql
@@CITYOBJECT/CITYOBJECT_GENERICATTRIB.sql
@@CITYOBJECT/CITYOBJECT_MEMBER.sql
@@CITYOBJECT/CITY_FURNITURE.sql
@@CITYOBJECT/EXTERNAL_REFERENCE.sql
@@CITYOBJECT/GENERALIZATION.sql
@@CITYOBJECT/Generic_CITYOBJECT.sql
@@CITYOBJECT/IMPLICIT_GEOMETRY.sql
@@CITYOBJECT/OBJECTCLASS.sql
@@CITYOBJECT/SURFACE_GEOMETRY.sql

@@CITYOBJECTGROUP/CITYOBJECTGROUP.sql
@@CITYOBJECTGROUP/GROUP_TO_CITYOBJECT.sql

@@BUILDING/ADDRESS.sql
@@BUILDING/ADDRESS_TO_BUILDING.sql
@@BUILDING/BUILDING.sql
@@BUILDING/BUILDING_FURNITURE.sql
@@BUILDING/BUILDING_INSTALLATION.sql
@@BUILDING/OPENING.sql
@@BUILDING/OPENING_TO_THEM_SURFACE.sql
@@BUILDING/ROOM.sql
@@BUILDING/THEMATIC_SURFACE.sql

@@APPEARANCE/APPEARANCE.sql
@@APPEARANCE/SURFACE_DATA.sql
@@APPEARANCE/TEXTUREPARAM.sql
@@APPEARANCE/APPEAR_TO_SURFACE_DATA.sql

@@DTM/BREAKLINE_RELIEF.sql
@@DTM/MASSPOINT_RELIEF.sql
@@DTM/RASTER_RELIEF.sql
@@DTM/RASTER_RELIEF_IMP.sql
@@DTM/RASTER_RELIEF_IMP_RDT.sql
@@DTM/RASTER_RELIEF_RDT.sql
@@DTM/RELIEF.sql
@@DTM/RELIEF_COMPONENT.sql
@@DTM/RELIEF_FEAT_TO_REL_COMP.sql
```

```

@@DTM/RELIEF FEATURE.sql
@@DTM/TIN_RELIEF.sql

@@ORTHOPHOTO/ORTHOPHOTO_RDT.sql;
@@ORTHOPHOTO/ORTHOPHOTO.sql;
@@ORTHOPHOTO/ORTHOPHOTO_RDT_IMP.sql;
@@ORTHOPHOTO/ORTHOPHOTO_IMP.sql;

@@TRANSPORTATION/TRANSPORTATION_COMPLEX.sql
@@TRANSPORTATION/TRAFFIC_AREA.sql

@@LANDUSE/LAND_USE.sql
@@LANDUSE/PLANT_COVER.sql
@@LANDUSE/SOLITARY_VEGETAT_OBJECT.sql

@@WATERBODY/WATERBODY.sql
@@WATERBODY/WATERBOD_TO_WATERBND_SRF.sql
@@WATERBODY/WATERBOUNDARY_SURFACE.sql

--// CREATE SEQUENCES
@@CITYOBJECT/CITYMODEL_SEQ.sql
@@CITYOBJECT/CITYOBJECT_GENERICATT_SEQ.sql
@@CITYOBJECT/CITYOBJECT_SEQ.sql
@@CITYOBJECT/EXTERNAL_REF_SEQ.sql
@@CITYOBJECT/IMPLICIT_GEOMETRY_SEQ.sql
@@CITYOBJECT/SURFACE_GEOMETRY_SEQ.sql

@@BUILDING/ADDRESS_SEQ.sql

@@APPEARANCE/APPEARANCE_SEQ.sql
@@APPEARANCE/SURFACE_DATA_SEQ.sql

@@ORTHOPHOTO/ORTHOPHOTO_SEQ.sql

--// Activate Constraints
@@ADD_CONSTRAINTS.sql

--// BUILD SIMPLE INDEXES
@@BUILD_SIMPLE_INDEX.sql

--// CREATE SPATIAL INDEX
@@SPATIAL_INDEX.sql

@@OBJECTCLASS_INSTANCES.sql
@@IMPORT_PROCEDURES.sql
@@DUMMY_IMPORT.sql

--// (possibly) activate versioning
BEGIN
  :VERSIONBATCHFILE := 'DO_NOTHING.sql';
END;
/
BEGIN
  IF ('&VERSIONING'='yes' OR '&VERSIONING'='YES' OR '&VERSIONING'='y' OR '&VERSIONING'='Y')
  THEN
    :VERSIONBATCHFILE := 'ENABLEVERSIONING.sql';
  END IF;
END;
/
-- Transfer the value from the bind variable to the substitution variable
column mc2 new value VERSIONBATCHFILE2 print
select :VERSIONBATCHFILE mc2 from dual;
@@&VERSIONBATCHFILE2

--// DML TRIGGER FOR RASTER TABLES
@@TRIGGER.sql;

@@MOSAIC.sql;

--// CREATE TABLES & PROCEDURES OF THE PLANNINGMANAGER
@@CREATE_PLANNINGMANAGER.sql

--// Tools and utilities
@@GEODB_REPORT.sql
@@GEODB_PKG/CREATE_GEODB_PKG.sql

SHOW ERRORS;
COMMIT;

SELECT 'DB creation complete!' as message from DUAL;

```

DATABASE_SRS.sql

```
-- DATABASE_SRS.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE DATABASE_SRS
(
  SRID NUMBER(38) NOT NULL,
  GML_SRS_NAME VARCHAR2(1000)
)
;
ALTER TABLE DATABASE_SRS
ADD CONSTRAINT DATABASE_SRS_PK PRIMARY KEY
(
  SRID
)
ENABLE
;
```


DO_NOTHING.sql

```
-- // empty SQL*Plus file
```

HINT_ON_MISSING_SRS.sql

```
-- HINT_ON_MISSING_SRS.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--              Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--              Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--              Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--              Viktor Stroh <stroh@ikg.uni-bonn.de>
--              Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2004-2007, Institute for Cartography and Geoinformation,
--              Universität Bonn, Germany
--              http://www.ikg.uni-bonn.de
--              (c) 2005-2007, lat/lon GmbH, Germany
--              http://www.lat-lon.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.1     | 2007-01-28 | release version | LPlu
--                                     | TKol
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
SET FEEDBACK OFF

prompt Your chosen SRID wasn't found in the MDSYS.CS_SRS table!
prompt If You want to use this db schema for storing the DHDN Soldner-Berlin data
prompt please execute as user SYS (with SYSDBA option)
prompt SQL script "SYSDBA\SOLDNER_BERLIN_SRS_10G_R2.sql"
```

OBJECTCLASS_INSTANCES.sql

```
-- OBJECTCLASS_INSTANCES.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                          Technische Universität Berlin, Germany
--                          http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
--
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     | GKoe
--                                     | CNag
--
DELETE FROM OBJECTCLASS;

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (0,'Undefined',NULL);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (1,'Object',NULL);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (2,'_AbstractFeature',1);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (3,'_CityObject',2);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (4,'LandUse',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (5,'GenericCityObject',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (6,'_VegetationObject',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (7,'SolitaryVegetationObject',6);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (8,'PlantCover',6);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (9,'WaterBody',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (10,'_WaterBoundarySurface',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (11,'WaterSurface',10);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (12,'WaterGroundSurface',10);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (13,'WaterClosureSurface',10);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (14,'ReliefFeature',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (15,'_ReliefComponent',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (16,'TINRelief',15);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
```

```
VALUES (17,'MassPointRelief',15);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (18,'BreaklineRelief',15);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (19,'Raster',15);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (20,'Orthophoto',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (21,'CityFurniture',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (22,'_TransportationObject',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (23,'CityObjectGroup',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (24,'_AbstractBuilding',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (25,'BuildingPart',24);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (26,'Building',24);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (27,'BuildingInstallation',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (28,'IntBuildingInstallation',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (29,'_BoundarySurface',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (30,'CeilingSurface',29);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (31,'InteriorWallSurface',29);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (32,'FloorSurface',29);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (33,'RoofSurface',29);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (34,'WallSurface',29);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (35,'GroundSurface',29);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (36,'ClosureSurface',29);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (37,'_Opening',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (38,'Window',37);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (39,'Door',37);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (40,'BuildingFurniture',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (41,'Room',3);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (42,'_TransportationComplex',22);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (43,'Track',42);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
```

```
VALUES (44,'Railway',42);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (45,'Road',42);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (46,'Square',42);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (47,'TrafficArea',22);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (48,'AuxiliaryTrafficArea',22);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (49,'FeatureCollection',2);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (50,'Appearance',2);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (51,'_SurfaceData',2);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (52,'_AbstractTexture',51);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (53,'X3DMaterial',51);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (54,'ParameterizedTexture',52);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (55,'GeoreferencedTexture',52);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (56,'TextureParametrization',1);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (57,'CityModel',49);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (58,'Address',2);

INSERT INTO OBJECTCLASS ( ID , CLASSNAME , SUPERCLASS_ID )
VALUES (59,'ImplicitGeometry',1);

COMMIT;
```

CITYOBJECT

CITYOBJECT.sql

```
-- CITYOBJECT.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     | GKoe
--                                     | CNag
--                                     | ASta
--
CREATE TABLE CITYOBJECT
(
  ID NUMBER NOT NULL,
  CLASS_ID NUMBER NOT NULL,
  GMLID VARCHAR2(256),
  GMLID_CODESPACE VARCHAR2(1000),
  ENVELOPE MDSYS.SDO_GEOMETRY,
  CREATION_DATE DATE NOT NULL,
  TERMINATION_DATE DATE,
  LAST_MODIFICATION_DATE DATE,
  UPDATING_PERSON VARCHAR2(256),
  REASON_FOR_UPDATE VARCHAR2(4000),
  LINEAGE VARCHAR2(256),
  XML_SOURCE CLOB
)
;
ALTER TABLE CITYOBJECT
ADD CONSTRAINT CITYOBJECT_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

CITYOBJECT_SEQ.sql

```
-- CITYMODEL_SEQ.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--
-- DROP SEQUENCE CITYMODEL_SEQ;
CREATE SEQUENCE CITYMODEL_SEQ INCREMENT BY 1 START WITH 1 MINVALUE 1 CACHE 10000;
```

CITYMODEL.sql

```
-- CITYMODEL.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE CITYMODEL
(
  ID NUMBER NOT NULL,
  GMLID VARCHAR2(256),
  GMLID_CODESPACE VARCHAR2(1000),
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  ENVELOPE MDSYS.SDO_GEOMETRY,
  CREATION_DATE DATE,
  TERMINATION_DATE DATE,
  LAST_MODIFICATION_DATE DATE,
  UPDATING_PERSON VARCHAR2(256),
  REASON_FOR_UPDATE VARCHAR2(4000),
  LINEAGE VARCHAR2(256)
)
;
ALTER TABLE CITYMODEL
ADD CONSTRAINT CITYMODEL_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```


CITYMODEL_SEQ.sql

```
-- CITYMODEL_SEQ.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--
-- DROP SEQUENCE CITYMODEL_SEQ;
CREATE SEQUENCE CITYMODEL_SEQ INCREMENT BY 1 START WITH 1 MINVALUE 1 CACHE 10000;
```

CITYOBJECT_MEMBER.sql

```
-- CITYOBJECT_MEMBER.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE CITYOBJECT_MEMBER
(
  CITYMODEL_ID NUMBER NOT NULL,
  CITYOBJECT_ID NUMBER NOT NULL
)
;
ALTER TABLE CITYOBJECT_MEMBER
ADD CONSTRAINT CITYOBJECT_MEMBER_PK PRIMARY KEY
(
  CITYMODEL_ID,
  CITYOBJECT_ID
)
ENABLE
;
```

EXTERNAL_REFERENCE.sql

```
-- EXTERNAL_REFERENCE.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE EXTERNAL_REFERENCE
(
  ID NUMBER NOT NULL,
  INFOSYS VARCHAR2(4000),
  NAME VARCHAR2(4000),
  URI VARCHAR2(4000),
  CITYOBJECT_ID NUMBER NOT NULL
)
;
ALTER TABLE EXTERNAL_REFERENCE
ADD CONSTRAINT EXTERNAL_REFERENCE_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

EXTERNAL_REF_SEQ.sql

```
-- EXTERNAL_REF_SEQ.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--
-- DROP SEQUENCE EXTERNAL_REF_SEQ;
CREATE SEQUENCE EXTERNAL_REF_SEQ INCREMENT BY 1 START WITH 1 MINVALUE 1 CACHE 10000;
```

GENERALIZATION.sql

```
-- GENERALIZATION.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE GENERALIZATION
(
  CITYOBJECT_ID NUMBER NOT NULL,
  GENERALIZES_TO_ID NUMBER NOT NULL
)
;
ALTER TABLE GENERALIZATION
ADD CONSTRAINT GENERALIZATION_PK PRIMARY KEY
(
  CITYOBJECT_ID,
  GENERALIZES_TO_ID
)
ENABLE
;
```

IMPLICIT_GEOMETRY.sql

```
-- IMPLICIT_GEOMETRY.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE IMPLICIT_GEOMETRY
(
  ID NUMBER NOT NULL,
  MIME_TYPE VARCHAR2(256),
  REFERENCE TO LIBRARY VARCHAR2(4000),
  LIBRARY_OBJECT BLOB,
  RELATIVE_GEOMETRY_ID NUMBER
)
;
ALTER TABLE IMPLICIT_GEOMETRY
ADD CONSTRAINT IMPLICIT_GEOMETRY_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

IMPLICIT_GEOMETRY_SEQ.sql

```
-- IMPLICIT_GEOMETRY_SEQ.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--
-- DROP SEQUENCE IMPLICIT_GEOMETRY_SEQ;
CREATE SEQUENCE IMPLICIT_GEOMETRY_SEQ INCREMENT BY 1 START WITH 1 MINVALUE 1 CACHE 10000;
```

OBJECTCLASS.sql

```
-- OBJECTCLASS.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE OBJECTCLASS
(
  ID NUMBER NOT NULL,
  CLASSNAME VARCHAR2(256),
  SUPERCLASS_ID NUMBER
)
;
ALTER TABLE OBJECTCLASS
ADD CONSTRAINT OBJECTCLASS_PK PRIMARY KEY
(
  ID
)
ENABLE;
```


GEOMETRY

SURFACE_GEOMETRY.sql

```
-- SURFACE_GEOMETRY.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASta
--
CREATE TABLE SURFACE_GEOMETRY
(
  ID NUMBER NOT NULL,
  GMLID VARCHAR2(256),
  GMLID_CODESPACE VARCHAR2(1000),
  PARENT_ID NUMBER,
  ROOT_ID NUMBER,
  IS_SOLID NUMBER(1, 0),
  IS_COMPOSITE NUMBER(1, 0),
  IS_TRIANGULATED NUMBER(1, 0),
  IS_XLINK NUMBER(1, 0),
  IS_REVERSE NUMBER(1, 0),
  GEOMETRY MDSYS.SDO_GEOMETRY
)
;
ALTER TABLE SURFACE_GEOMETRY
ADD CONSTRAINT SURFACE_GEOMETRY_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

SURFACE_GEOMETRY_SEQ.sql

```
-- SURFACE_GEOMETRY_SEQ.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description                | Author
-- 2.0.0   | 2007-11-23 | release version            | TKol
--                                     | GKoe
--                                     | CNag
--
-- DROP SEQUENCE SURFACE_GEOMETRY_SEQ;
CREATE SEQUENCE SURFACE_GEOMETRY_SEQ INCREMENT BY 1 START WITH 1 MINVALUE 1 CACHE 10000;
```

APPEARANCE

APPEARANCE.sql

```
-- APPEARANCE.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASta
--
CREATE TABLE APPEARANCE
(
  ID NUMBER NOT NULL,
  GMLID VARCHAR2(256),
  GMLID_CODESPACE VARCHAR2(1000),
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  THEME VARCHAR2(256),
  CITYMODEL_ID NUMBER,
  CITYOBJECT_ID NUMBER
)
;
ALTER TABLE APPEARANCE
ADD CONSTRAINT APPEARANCE_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

APPEARANCE_SEQ.sql

```
-- APPEARANCE_SEQ.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--
-- DROP SEQUENCE APPEARANCE_SEQ;
CREATE SEQUENCE APPEARANCE_SEQ INCREMENT BY 1 START WITH 1 MINVALUE 1 CACHE 10000;
```

APPEAR_TO_SURFACE_DATA.sql

```
-- APPEAR_TO_SURFACE_DATA.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE APPEAR_TO_SURFACE_DATA
(
  SURFACE_DATA_ID NUMBER NOT NULL,
  APPEARANCE_ID NUMBER NOT NULL
)
;
ALTER TABLE APPEAR_TO_SURFACE_DATA
ADD CONSTRAINT APPEARANCE_TO_SURFACEIDAT_PK PRIMARY KEY
(
  SURFACE_DATA_ID,
  APPEARANCE_ID
)
ENABLE
;
```

SURFACE_DATA.sql

```
-- SURFACE_DATA.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
--
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     Asta
--
CREATE TABLE SURFACE_DATA
(
  ID NUMBER NOT NULL,
  GMLID VARCHAR2(256),
  GMLID_CODESPACE VARCHAR2(1000),
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  IS_FRONT NUMBER(1, 0),
  TYPE VARCHAR2(30),
  X3D_SHININESS BINARY DOUBLE,
  X3D_TRANSPARENCY BINARY DOUBLE,
  X3D_AMBIENT_INTENSITY BINARY DOUBLE,
  X3D_SPECULAR_COLOR VARCHAR2(256),
  X3D_DIFFUSE_COLOR VARCHAR2(256),
  X3D_EMISSIVE_COLOR VARCHAR2(256),
  X3D_IS_SMOOTH NUMBER(1, 0),
  TEX_IMAGE_URI VARCHAR2(4000),
  TEX_IMAGE ORDSYS.ORDIMAGE,
  TEX_MIME_TYPE VARCHAR2(256),
  TEX_TEXTURE_TYPE VARCHAR2(256),
  TEX_WRAP_MODE VARCHAR2(256),
  TEX_BORDER_COLOR VARCHAR2(256),
  GT_PREFER_WORLDFILE NUMBER(1, 0),
  GT_ORIENTATION VARCHAR2(256),
  GT_REFERENCE_POINT MDSYS.SDO_GEOMETRY
)
;
ALTER TABLE SURFACE_DATA
ADD CONSTRAINT SURFACE_DATA_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

SURFACE_DATA_SEQ.sql

```
-- SURFACE_DATA_SEQ.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--
-- DROP SEQUENCE SURFACE_DATA_SEQ;
CREATE SEQUENCE SURFACE_DATA_SEQ INCREMENT BY 1 START WITH 1 MINVALUE 1 CACHE 10000;
```

TEXTUREPARAM.sql

```
-- TEXTUREPARAM.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE TEXTUREPARAM
(
  SURFACE_GEOMETRY_ID NUMBER NOT NULL,
  IS_TEXTURE_PARAMETRIZATION NUMBER(1, 0),
  WORLD_TO_TEXTURE VARCHAR2(1000),
  TEXTURE_COORDINATES VARCHAR2(4000),
  SURFACE_DATA_ID NUMBER NOT NULL
)
;
ALTER TABLE TEXTUREPARAM
ADD CONSTRAINT TEXTUREPARAM_PK PRIMARY KEY
(
  SURFACE_GEOMETRY_ID,
  SURFACE_DATA_ID
)
ENABLE
;
```


BUILDING

BUILDING.sql

```
-- BUILDING.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASta
--
CREATE TABLE BUILDING
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  BUILDING_PARENT_ID NUMBER,
  BUILDING_ROOT_ID NUMBER,
  DESCRIPTION VARCHAR2(4000),
  CLASS VARCHAR2(256),
  FUNCTION VARCHAR2(1000),
  USAGE VARCHAR2(1000),
  YEAR_OF_CONSTRUCTION DATE,
  YEAR_OF_DEMOLITION DATE,
  ROOF_TYPE VARCHAR2(256),
  MEASURED_HEIGHT BINARY_DOUBLE,
  STOREYS_ABOVE_GROUND NUMBER(8),
  STOREYS_BELOW_GROUND NUMBER(8),
  STOREY_HEIGHTS_ABOVE_GROUND VARCHAR2(4000),
  STOREY_HEIGHTS_BELOW_GROUND VARCHAR2(4000),
  LOD1_TERRAIN_INTERSECTION MDSYS.SDO_GEOMETRY,
  LOD2_TERRAIN_INTERSECTION MDSYS.SDO_GEOMETRY,
  LOD3_TERRAIN_INTERSECTION MDSYS.SDO_GEOMETRY,
  LOD4_TERRAIN_INTERSECTION MDSYS.SDO_GEOMETRY,
  LOD2_MULTI_CURVE MDSYS.SDO_GEOMETRY,
  LOD3_MULTI_CURVE MDSYS.SDO_GEOMETRY,
  LOD4_MULTI_CURVE MDSYS.SDO_GEOMETRY,
  LOD1_GEOMETRY_ID NUMBER,
  LOD2_GEOMETRY_ID NUMBER,
  LOD3_GEOMETRY_ID NUMBER,
  LOD4_GEOMETRY_ID NUMBER
)
;
ALTER TABLE BUILDING
ADD CONSTRAINT BUILDING_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

ADDRESS.sql

```
-- ADDRESS.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     | GKoe
--                                     | CNag
--                                     | ASa
--
CREATE TABLE ADDRESS
(
  ID NUMBER NOT NULL,
  STREET VARCHAR2(1000),
  HOUSE_NUMBER VARCHAR2(256),
  PO_BOX VARCHAR2(256),
  ZIP_CODE VARCHAR2(256),
  CITY VARCHAR2(256),
  STATE VARCHAR2(256),
  COUNTRY VARCHAR2(256),
  MULTI_POINT MDSYS.SDO_GEOMETRY,
  XAL_SOURCE CLOB
)
;

ALTER TABLE ADDRESS
ADD CONSTRAINT ADDRESS_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

ADDRESS_SEQ.sql

```
-- ADDRESS_SEQ.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description                | Author
-- 2.0.0   | 2007-11-23 | release version            | TKol
--                                     | GKoe
--                                     | CNag
--
-- DROP SEQUENCE ADDRESS_SEQ;
CREATE SEQUENCE ADDRESS_SEQ INCREMENT BY 1 START WITH 1 MINVALUE 1 CACHE 10000;
```

ADDRESS_TO_BUILDING.sql

```
-- ADDRESS_TO_BUILDING.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE ADDRESS_TO_BUILDING
(
  BUILDING_ID NUMBER NOT NULL,
  ADDRESS_ID NUMBER NOT NULL
)
;
ALTER TABLE ADDRESS_TO_BUILDING
ADD CONSTRAINT ADDRESS_TO_BUILDING_PK PRIMARY KEY
(
  BUILDING_ID,
  ADDRESS_ID
)
ENABLE
;
```

BUILDING_FURNITURE.sql

```
-- BUILDING_FURNITURE.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE BUILDING_FURNITURE
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  CLASS VARCHAR2(256),
  FUNCTION VARCHAR2(1000),
  USAGE VARCHAR2(1000),
  ROOM_ID NUMBER NOT NULL,
  LOD4_GEOMETRY_ID NUMBER,
  LOD4_IMPLICIT_REP_ID NUMBER,
  LOD4_IMPLICIT_REF_POINT MDSYS.SDO_GEOMETRY,
  LOD4_IMPLICIT_TRANSFORMATION VARCHAR2(1000)
)
;
ALTER TABLE BUILDING_FURNITURE
ADD CONSTRAINT BUILDING_FURNITURE_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

BUILDING_INSTALLATION.sql

```
-- BUILDING_INSTALLATION.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE BUILDING_INSTALLATION
(
  ID NUMBER NOT NULL,
  IS_EXTERNAL NUMBER(1, 0),
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  CLASS VARCHAR2(256),
  FUNCTION VARCHAR2(1000),
  USAGE VARCHAR2(1000),
  BUILDING_ID NUMBER,
  ROOM_ID NUMBER,
  LOD2_GEOMETRY_ID NUMBER,
  LOD3_GEOMETRY_ID NUMBER,
  LOD4_GEOMETRY_ID NUMBER
)
;
ALTER TABLE BUILDING_INSTALLATION
ADD CONSTRAINT BUILDING_INSTALLATION_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

OPENING.sql

```
-- OPENING.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE OPENING
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  TYPE VARCHAR2(256),
  ADDRESS ID NUMBER,
  LOD3_MULTI_SURFACE_ID NUMBER,
  LOD4_MULTI_SURFACE_ID NUMBER
)
;
ALTER TABLE OPENING
ADD CONSTRAINT OPENING_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

OPENING_TO_THEM_SURFACE.sql

```
-- OPENING_TO_THEM_SURFACE.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE OPENING_TO_THEM_SURFACE
(
  OPENING_ID NUMBER NOT NULL,
  THEMATIC_SURFACE_ID NUMBER NOT NULL
)
;
ALTER TABLE OPENING_TO_THEM_SURFACE
ADD CONSTRAINT OPENING_TO_THEM_SURFACE_PK PRIMARY KEY
(
  OPENING_ID,
  THEMATIC_SURFACE_ID
)
ENABLE
;
```


ROOM.sql

```
-- ROOM.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE ROOM
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  CLASS VARCHAR2(256),
  FUNCTION VARCHAR2(1000),
  USAGE VARCHAR2(1000),
  BUILDING_ID NUMBER NOT NULL,
  LOD4_GEOMETRY_ID NUMBER
)
;
ALTER TABLE ROOM
ADD CONSTRAINT ROOM_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

THEMATIC_SURFACE.sql

```
-- THEMATIC_SURFACE.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
--
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE THEMATIC_SURFACE
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  TYPE VARCHAR2(256),
  BUILDING_ID NUMBER,
  ROOM_ID NUMBER,
  LOD2_MULTI_SURFACE_ID NUMBER,
  LOD3_MULTI_SURFACE_ID NUMBER,
  LOD4_MULTI_SURFACE_ID NUMBER
)
;
ALTER TABLE THEMATIC_SURFACE
ADD CONSTRAINT THEMATIC_SURFACE_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

CITYFURNITURE

CITY_FURNITURE.sql

```
-- CITY_FURNITURE.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
--
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     | GKoe
--                                     | CNag
--                                     | ASta
--
CREATE TABLE CITY_FURNITURE
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  CLASS VARCHAR2(256),
  FUNCTION VARCHAR2(1000),
  LOD1_TERRAIN_INTERSECTION MDSYS.SDO_GEOMETRY,
  LOD2_TERRAIN_INTERSECTION MDSYS.SDO_GEOMETRY,
  LOD3_TERRAIN_INTERSECTION MDSYS.SDO_GEOMETRY,
  LOD4_TERRAIN_INTERSECTION MDSYS.SDO_GEOMETRY,
  LOD1_GEOMETRY_ID NUMBER,
  LOD2_GEOMETRY_ID NUMBER,
  LOD3_GEOMETRY_ID NUMBER,
  LOD4_GEOMETRY_ID NUMBER,
  LOD1_IMPLICIT_REP_ID NUMBER,
  LOD2_IMPLICIT_REP_ID NUMBER,
  LOD3_IMPLICIT_REP_ID NUMBER,
  LOD4_IMPLICIT_REP_ID NUMBER,
  LOD1_IMPLICIT_REF_POINT MDSYS.SDO_GEOMETRY,
  LOD2_IMPLICIT_REF_POINT MDSYS.SDO_GEOMETRY,
  LOD3_IMPLICIT_REF_POINT MDSYS.SDO_GEOMETRY,
  LOD4_IMPLICIT_REF_POINT MDSYS.SDO_GEOMETRY,
  LOD1_IMPLICIT_TRANSFORMATION VARCHAR2(1000),
  LOD2_IMPLICIT_TRANSFORMATION VARCHAR2(1000),
  LOD3_IMPLICIT_TRANSFORMATION VARCHAR2(1000),
  LOD4_IMPLICIT_TRANSFORMATION VARCHAR2(1000)
);
ALTER TABLE CITY_FURNITURE
ADD CONSTRAINT CITY_FURNITURE_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

CITYOBJECTGROUP

CITYOBJECTGROUP.sql

```
-- CITYOBJECTGROUP.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                          Technische Universität Berlin, Germany
--                          http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     | GKoe
--                                     | CNag
--                                     | ASta
--
CREATE TABLE CITYOBJECTGROUP
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  CLASS VARCHAR2(256),
  FUNCTION VARCHAR2(1000),
  USAGE VARCHAR2(1000),
  GEOMETRY MDSYS.SDO_GEOMETRY,
  SURFACE_GEOMETRY_ID NUMBER,
  PARENT_CITYOBJECT_ID NUMBER
)
;
ALTER TABLE CITYOBJECTGROUP
ADD CONSTRAINT CITYOBJECTGROUP_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

GROUP_TO_CITYOBJECT.sql

```
-- GROUP_TO_CITYOBJECT.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE GROUP_TO_CITYOBJECT
(
  CITYOBJECT_ID NUMBER NOT NULL,
  CITYOBJECTGROUP_ID NUMBER NOT NULL,
  ROLE VARCHAR2(256)
)
;
ALTER TABLE GROUP_TO_CITYOBJECT
ADD CONSTRAINT GROUP_TO_CITYOBJECT_PK PRIMARY KEY
(
  CITYOBJECT_ID,
  CITYOBJECTGROUP_ID
)
ENABLE
;
```

DTM**BREAKLINE_RELIEF.sql**

```
-- BREAKLINE_RELIEF.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     | GKoe
--                                     | CNag
--                                     | ASta
--
CREATE TABLE BREAKLINE_RELIEF
(
  ID NUMBER NOT NULL,
  RIDGE_OR_VALLEY_LINES MDSYS.SDO_GEOMETRY,
  BREAK_LINES MDSYS.SDO_GEOMETRY
)
;
ALTER TABLE BREAKLINE_RELIEF
ADD CONSTRAINT BREAKLINE_RELIEF_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

DTM_SEQ.sql

```
-- DTM_SEQ.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--               Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--               Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--               Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--               Viktor Stroh <stroh@ikg.uni-bonn.de>
--               Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2004-2006, Institute for Cartography and Geoinformation,
--               Universität Bonn, Germany
--               http://www.ikg.uni-bonn.de
--               (c) 2005-2006, lat/lon GmbH, Germany
--               http://www.lat-lon.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
--
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.0      | 2006-04-03 | release version | LPlu
--                                     | TKol
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
-- DROP SEQUENCE "RASTER_REL_IMP_SEQ";
CREATE SEQUENCE "RASTER_REL_IMP_SEQ" INCREMENT BY 1 START WITH 1 MINVALUE 1;
--
-- DROP SEQUENCE "RASTER_REL_RDT_SEQ";
CREATE SEQUENCE "RASTER_REL_RDT_SEQ" INCREMENT BY 1 START WITH 1 MINVALUE 1;
--
-- DROP SEQUENCE "RASTER_REL_RDT_IMP_SEQ";
CREATE SEQUENCE "RASTER_REL_RDT_IMP_SEQ" INCREMENT BY 1 START WITH 1 MINVALUE 1;
```

MASSPOINT_RELIEF.sql

```
-- MASSPOINT_RELIEF.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE MASSPOINT_RELIEF
(
  ID NUMBER NOT NULL,
  RELIEF_POINTS MDSYS.SDO_GEOMETRY
)
;
ALTER TABLE MASSPOINT_RELIEF
ADD CONSTRAINT MASSPOINT_RELIEF_PK PRIMARY KEY
(
  ID
)
ENABLE
;

```


RASTER_RELIEF.sql

```
-- RASTER_RELIEF.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
--               Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--               Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--               Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--               Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--               Viktor Stroh <stroh@ikg.uni-bonn.de>
--               Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--               (c) 2004-2006,  Institute for Cartography and Geoinformation,
--                               Universität Bonn, Germany
--                               http://www.ikg.uni-bonn.de
--               (c) 2005-2006,  lat/lon GmbH, Germany
--                               http://www.lat-lon.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
--
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | LPlu
--                                     TKol
--                                     GGro
--                                     JSch
--                                     VStr
--                                     APot
--
-- DROP TABLE "RASTER_RELIEF" CASCADE CONSTRAINT PURGE;
--
CREATE TABLE "RASTER_RELIEF" (
  "ID" NUMBER NOT NULL,
  -- "LOD" NUMBER (1) NOT NULL,
  "RASTERPROPERTY" MDSYS.SDO_GEORASTER NOT NULL
  -- "RELIEF_ID" NUMBER NOT NULL,
  -- "NAME" VARCHAR2 (256),
  -- "TYPE" VARCHAR2 (256),
  -- "EXTENT" MDSYS.SDO_GEOMETRY
);
--
ALTER TABLE "RASTER_RELIEF"
ADD CONSTRAINT "RASTER_RLF_PK" PRIMARY KEY ( "ID" ) ENABLE;
```

RASTER_RELIEF_IMP.sql

```
-- RASTER_RELIEF_IMP.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--              Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--              Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--              Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--              Viktor Stroh <stroh@ikg.uni-bonn.de>
--              Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2004-2006, Institute for Cartography and Geoinformation,
--              Universität Bonn, Germany
--              http://www.ikg.uni-bonn.de
--              (c) 2005-2006, lat/lon GmbH, Germany
--              http://www.lat-lon.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.0     | 2006-04-03 | release version | LPlu
--                                     | TKol
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
-- DROP TABLE "RASTER_RELIEF_IMP" CASCADE CONSTRAINT PURGE;

CREATE TABLE "RASTER_RELIEF_IMP" (
  "ID" NUMBER NOT NULL,
  "RASTERPROPERTY" MDSYS.SDO_GEORASTER,
  "RELIEF_ID" NUMBER,
  "RASTER_RELIEF_ID" NUMBER,
  "FILENAME" VARCHAR2 (4000),
  "FOOTPRINT" MDSYS.SDO_GEOMETRY );

ALTER TABLE "RASTER_RELIEF_IMP"
ADD CONSTRAINT "RASTER_RLF_IMP_PK" PRIMARY KEY ( "ID" ) ENABLE;
```

RASTER_RELIEF_IMP_RDT.sql

```
-- RASTER_RELIEF_IMP_RDT.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--              Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--              Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--              Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--              Viktor Stroh <stroh@ikg.uni-bonn.de>
--              Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2004-2006, Institute for Cartography and Geoinformation,
--              Universität Bonn, Germany
--              http://www.ikg.uni-bonn.de
--              (c) 2005-2006, lat/lon GmbH, Germany
--              http://www.lat-lon.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
--
-----
-- About:
--
-- must be created after raster table(s) has been created
--
-----
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.0     | 2006-04-03 | release version | LPlu
--                                     | TKol
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
--
-- DROP TABLE "RASTER_RELIEF_IMP_RDT" CASCADE CONSTRAINT PURGE;
--
CREATE TABLE "RASTER_RELIEF_IMP_RDT" OF SDO_RASTER(
PRIMARY KEY ( RASTERID, PYRAMIDLEVEL, BANDBLOCKNUMBER, ROWBLOCKNUMBER, COLUMNBLOCKNUMBER ) )
LOB(RASTERBLOCK) STORE AS (NOCACHE NOLOGGING);
```

RASTER_RELIEF_RDT.sql

```
-- RASTER_RELIEF_RDT.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--               Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--               Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--               Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--               Viktor Stroh <stroh@ikg.uni-bonn.de>
--               Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2004-2006, Institute for Cartography and Geoinformation,
--               Universität Bonn, Germany
--               http://www.ikg.uni-bonn.de
--               (c) 2005-2006, lat/lon GmbH, Germany
--               http://www.lat-lon.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-- must be created after raster table(s) has been created
-----
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.0     | 2006-04-03 | release version | LPlu
--                                     | TKol
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
-- DROP TABLE "RASTER_RELIEF_RDT" CASCADE CONSTRAINT PURGE;

CREATE TABLE "RASTER_RELIEF_RDT" OF SDO_RASTER(
  PRIMARY KEY ( RASTERID, PYRAMIDLEVEL, BÄNDBLOCKNUMBER, ROWBLOCKNUMBER, COLUMNBLOCKNUMBER ) )
LOB(RASTERBLOCK) STORE AS (NOCACHE NOLOGGING);
```

RASTER_RELIEF_RDT_ID_TRIGGER.sql

```
-- RASTER_RELIEF_RDT_ID_TRIGGER.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--              Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--              Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--              Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--              Viktor Stroh <stroh@ikg.uni-bonn.de>
--              Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2004-2006, Institute for Cartography and Geoinformation,
--              Universität Bonn, Germany
--              http://www.ikg.uni-bonn.de
--              (c) 2005-2006, lat/lon GmbH, Germany
--              http://www.lat-lon.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.0     | 2006-04-03 | release version | LPlu
--                                     | TKol
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
EXECUTE DBMS_WM.BeginDDL('RASTER_RELIEF_RDT');

drop trigger RASTER_RELIEF_RDT_ID;

create trigger RASTER_RELIEF_RDT_ID
before insert on RASTER_RELIEF_RDT_LTS
for each row
begin
    if :new.id is null then
        select RASTER_RELIEF_RDT_SEQ.nextval into :new.id from dual;
    end if;
end;
/

EXECUTE DBMS_WM.CommitDDL('RASTER_RELIEF_RDT');
```

RASTER_RELIEF_RDT_IMP_ID_TRIGGER.sql

```
-- RASTER_RELIEF_RDT_IMP_ID_TRIGGER.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--              Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--              Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--              Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--              Viktor Stroh <stroh@ikg.uni-bonn.de>
--              Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2004-2006, Institute for Cartography and Geoinformation,
--              Universität Bonn, Germany
--              http://www.ikg.uni-bonn.de
--              (c) 2005-2006, lat/lon GmbH, Germany
--              http://www.lat-lon.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.0     | 2006-04-03 | release version | LPlu
--                                     | TKol
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
EXECUTE DBMS_WM.BeginDDL('RASTER_RELIEF_RDT_IMP');

drop trigger RASTER_RELIEF_RDT_IMP_ID;

create trigger RASTER_RELIEF_RDT_IMP_ID
before insert on RASTER_RELIEF_RDT_IMP_LTS
for each row
begin
    if :new.id is null then
        select RASTER_RELIEF_RDT_IMP_SEQ.nextval into :new.id from dual;
    end if;
end;
/

EXECUTE DBMS_WM.CommitDDL('RASTER_RELIEF_RDT_IMP');
```

RELIEF.sql

```
-- RELIEF.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--              Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--              Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--              Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--              Viktor Stroh <stroh@ikg.uni-bonn.de>
--              Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2004-2006, Institute for Cartography and Geoinformation,
--              Universität Bonn, Germany
--              http://www.ikg.uni-bonn.de
--              (c) 2005-2006, lat/lon GmbH, Germany
--              http://www.lat-lon.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.0     | 2006-04-03 | release version | LPlu
--                                     | TKol
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
-- DROP TABLE "RELIEF" CASCADE CONSTRAINT PURGE;

CREATE TABLE "RELIEF" (
  "ID" NUMBER NOT NULL,
  "NAME" VARCHAR2 (256),
  "TYPE" VARCHAR2 (256),
  "LODGROUP" NUMBER (1) NOT NULL );

ALTER TABLE "RELIEF"
ADD CONSTRAINT "RELIEF_PK" PRIMARY KEY ( "ID" ) ENABLE;
```

RELIEF_COMPONENT.sql

```
-- RELIEF_COMPONENT.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE RELIEF_COMPONENT
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  LOD NUMBER(1),
  EXTENT MDSYS.SDO_GEOMETRY
)
;
ALTER TABLE RELIEF_COMPONENT
ADD CONSTRAINT RELIEF_COMPONENT_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```


RELIEF_FEATURE.sql

```
-- RELIEF_FEATURE.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE RELIEF_FEATURE
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  LOD NUMBER(1)
)
;
ALTER TABLE RELIEF_FEATURE
ADD CONSTRAINT RELIEF_FEATURE_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

RELIEF_FEAT_TO_REL_COMP.sql

```
-- RELIEF_FEAT_TO_REL_COMP.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE RELIEF_FEAT_TO_REL_COMP
(
RELIEF_COMPONENT_ID NUMBER NOT NULL,
RELIEF_FEATURE_ID NUMBER NOT NULL
)
;
ALTER TABLE RELIEF_FEAT_TO_REL_COMP
ADD CONSTRAINT RELIEF_FEATURE_TO_RELIEF__PK PRIMARY KEY
(
RELIEF_COMPONENT_ID,
RELIEF_FEATURE_ID
)
ENABLE
;
```

TIN_RELIEF.sql

```
-- TIN_RELIEF.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--
-- DROP TABLE TIN_RELIEF CASCADE CONSTRAINT PURGE;
ASta

CREATE TABLE TIN_RELIEF
(
  ID NUMBER NOT NULL,
  MAX_LENGTH BINARY_DOUBLE,
  STOP_LINES MDSYS.SDO_GEOMETRY,
  BREAK_LINES MDSYS.SDO_GEOMETRY,
  CONTROL_POINTS MDSYS.SDO_GEOMETRY,
  SURFACE_GEOMETRY_ID NUMBER
)
;
ALTER TABLE TIN_RELIEF
ADD CONSTRAINT TIN_RELIEF_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

GENERICICS

CITYOBJECT_GENERICATTRIB.sql

```
-- CITYOBJECT_GENERICATTRIB.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
--
-----
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     | GKoe
--                                     | CNag
--                                     | ASta
--
CREATE TABLE CITYOBJECT_GENERICATTRIB
(
  ID NUMBER NOT NULL,
  ATTRNAME VARCHAR2(256) NOT NULL,
  DATATYPE NUMBER(1),
  STRVAL VARCHAR2(4000),
  INTVAL NUMBER,
  REALVAL NUMBER,
  URIVAL VARCHAR2(4000),
  DATEVAL DATE,
  GEOMVAL MDSYS.SDO_GEOMETRY,
  BLOBVAL BLOB,
  CITYOBJECT_ID NUMBER NOT NULL,
  SURFACE_GEOMETRY_ID NUMBER
)
;
ALTER TABLE CITYOBJECT_GENERICATTRIB
ADD CONSTRAINT CITYOBJECT_GENERICATTRIB_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

CITYOBJECT_GENERICATT_SEQ.sql

```
-- CITYOBJECT_GENERICATT_SEQ.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     | GKoe
--                                     | CNag
--
-- DROP SEQUENCE CITYOBJECT_GENERICATT_SEQ;
CREATE SEQUENCE CITYOBJECT_GENERICATT_SEQ INCREMENT BY 1 START WITH 1 MINVALUE 1 CACHE 10000;
```

GENERIC_CITYOBJECT.sql

```
-- GENERIC_CITYOBJECT.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
--
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     Asta
--
CREATE TABLE GENERIC_CITYOBJECT
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  CLASS VARCHAR2(256),
  FUNCTION VARCHAR2(1000),
  USAGE VARCHAR2(1000),
  LOD0_TERRAIN_INTERSECTION MDSYS.SDO_GEOMETRY,
  LOD1_TERRAIN_INTERSECTION MDSYS.SDO_GEOMETRY,
  LOD2_TERRAIN_INTERSECTION MDSYS.SDO_GEOMETRY,
  LOD3_TERRAIN_INTERSECTION MDSYS.SDO_GEOMETRY,
  LOD4_TERRAIN_INTERSECTION MDSYS.SDO_GEOMETRY,
  LOD0_GEOMETRY_ID NUMBER,
  LOD1_GEOMETRY_ID NUMBER,
  LOD2_GEOMETRY_ID NUMBER,
  LOD3_GEOMETRY_ID NUMBER,
  LOD4_GEOMETRY_ID NUMBER,
  LOD0_IMPLICIT_REP_ID NUMBER,
  LOD1_IMPLICIT_REP_ID NUMBER,
  LOD2_IMPLICIT_REP_ID NUMBER,
  LOD3_IMPLICIT_REP_ID NUMBER,
  LOD4_IMPLICIT_REP_ID NUMBER,
  LOD0_IMPLICIT_REF_POINT MDSYS.SDO_GEOMETRY,
  LOD1_IMPLICIT_REF_POINT MDSYS.SDO_GEOMETRY,
  LOD2_IMPLICIT_REF_POINT MDSYS.SDO_GEOMETRY,
  LOD3_IMPLICIT_REF_POINT MDSYS.SDO_GEOMETRY,
  LOD4_IMPLICIT_REF_POINT MDSYS.SDO_GEOMETRY,
  LOD0_IMPLICIT_TRANSFORMATION VARCHAR2(1000),
  LOD1_IMPLICIT_TRANSFORMATION VARCHAR2(1000),
  LOD2_IMPLICIT_TRANSFORMATION VARCHAR2(1000),
  LOD3_IMPLICIT_TRANSFORMATION VARCHAR2(1000),
  LOD4_IMPLICIT_TRANSFORMATION VARCHAR2(1000)
);
ALTER TABLE GENERIC_CITYOBJECT
ADD CONSTRAINT GENERIC_CITYOBJECT_PK PRIMARY KEY
(
  ID
)
ENABLE;
```

LANDUSE

LAND_USE.sql

```
-- LAND_USE.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     | GKoe
--                                     | CNag
--                                     | ASta
--
CREATE TABLE LAND_USE
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  CLASS VARCHAR2(256),
  FUNCTION VARCHAR2(1000),
  USAGE VARCHAR2(1000),
  LOD0_MULTI_SURFACE_ID NUMBER,
  LOD1_MULTI_SURFACE_ID NUMBER,
  LOD2_MULTI_SURFACE_ID NUMBER,
  LOD3_MULTI_SURFACE_ID NUMBER,
  LOD4_MULTI_SURFACE_ID NUMBER
)
;
ALTER TABLE LAND_USE
ADD CONSTRAINT LAND_USE_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

ORTHOPHOTO

ORTHOPHOTO.sql

```
-- ORTHOPHOTO.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--              Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--              Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--              Viktor Stroh <stroh@ikg.uni-bonn.de>
--              Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2007,      Institute for Geodesy and Geoinformation Science,
--              Technische Universität Berlin, Germany
--              http://www.igg.tu-berlin.de
--              (c) 2004-2006, Institute for Cartography and Geoinformation,
--              Universität Bonn, Germany
--              http://www.ikg.uni-bonn.de
--              (c) 2005-2006, lat/lon GmbH, Germany
--              http://www.lat-lon.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
--
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0     | 2007-12-10 | release version | TKol
--                                     | LPlu
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
-- DROP TABLE "ORTHOPHOTO" CASCADE CONSTRAINT PURGE;
CREATE TABLE "ORTHOPHOTO" (
  "ID" NUMBER NOT NULL,
  "LOD" NUMBER (1) NOT NULL,
  "NAME" VARCHAR2 (256),
  "TYPE" VARCHAR2 (256),
  "DATUM" DATE,
  "ORTHOPHOTOPROPERTY" MDSYS.SDO_GEORASTER );
ALTER TABLE "ORTHOPHOTO"
ADD CONSTRAINT "ORTHOPHOTO_PK" PRIMARY KEY ( "ID" ) ENABLE;
```


ORTHOPHOTO_IMP.sql

```
-- ORTHOPHOTO_IMP.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--               Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--               Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--               Viktor Stroh <stroh@ikg.uni-bonn.de>
--               Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2007,      Institute for Geodesy and Geoinformation Science,
--               Technische Universität Berlin, Germany
--               http://www.igg.tu-berlin.de
--               (c) 2004-2006, Institute for Cartography and Geoinformation,
--               Universität Bonn, Germany
--               http://www.ikg.uni-bonn.de
--               (c) 2005-2006, lat/lon GmbH, Germany
--               http://www.lat-lon.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
--
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0     | 2007-12-10 | release version | TKol
--                                     | LPlu
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
-- DROP TABLE "ORTHOPHOTO_IMP" CASCADE CONSTRAINT PURGE;
--
-- CREATE TABLE "ORTHOPHOTO_IMP" (
--   "ID" NUMBER NOT NULL,
--   "ORTHOPHOTOPROPERTY" MDSYS.SDO_GEORASTER,
--   "FILENAME" VARCHAR2 (4000),
--   "FOOTPRINT" MDSYS.SDO_GEOMETRY );
--
-- ALTER TABLE "ORTHOPHOTO_IMP"
-- ADD CONSTRAINT "ORTHOPHOTO_IMP_PK" PRIMARY KEY ( "ID" ) ENABLE;
```

ORTHOPHOTO_RDT.sql

```
-- ORTHOPHOTO_RDT.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--               Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--               Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--               Viktor Stroh <stroh@ikg.uni-bonn.de>
--               Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2007,      Institute for Geodesy and Geoinformation Science,
--               Technische Universität Berlin, Germany
--               http://www.igg.tu-berlin.de
--               (c) 2004-2006, Institute for Cartography and Geoinformation,
--               Universität Bonn, Germany
--               http://www.ikg.uni-bonn.de
--               (c) 2005-2006, lat/lon GmbH, Germany
--               http://www.lat-lon.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-- must be created after raster table(s) has been created
-----
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0     | 2007-12-10 | release version | TKol
--                                     | LPlu
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
-- DROP TABLE "ORTHOPHOTO_RDT" CASCADE CONSTRAINT PURGE;

CREATE TABLE "ORTHOPHOTO_RDT" OF SDO_RASTER(
  PRIMARY KEY ( RASTERID, PYRAMIDLEVEL, BANDBLOCKNUMBER, ROWBLOCKNUMBER, COLUMNBLOCKNUMBER )
) LOB(RASTERBLOCK) STORE AS (NOCACHE NOLOGGING);
```

ORTHOPHOTO_RDT_ID_TRIGGER.sql

```
-- ORTHOPHOTO_RDT_ID_TRIGGER.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--               Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--               Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--               Viktor Stroh <stroh@ikg.uni-bonn.de>
--               Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2007,      Institute for Geodesy and Geoinformation Science,
--               Technische Universität Berlin, Germany
--               http://www.igg.tu-berlin.de
--               (c) 2004-2006, Institute for Cartography and Geoinformation,
--               Universität Bonn, Germany
--               http://www.ikg.uni-bonn.de
--               (c) 2005-2006, lat/lon GmbH, Germany
--               http://www.lat-lon.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0     | 2007-12-10 | release version | TKol
--                                     | LPlu
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
drop trigger ORTHO_RDT_TRG;

create trigger ORTHO_RDT_TRG
before insert on ORTHOPHOTO_RDT
for each row
begin
    if :new.id is null then
        select ORTHOPHOTO_RDT_SEQ.nextval into :new.id from dual;
    end if;
end;
/
```

ORTHOPHOTO_RDT_IMP.sql

```
-- ORTHOPHOTO_RDT_IMP.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--               Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--               Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--               Viktor Stroh <stroh@ikg.uni-bonn.de>
--               Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2007,      Institute for Geodesy and Geoinformation Science,
--               Technische Universität Berlin, Germany
--               http://www.igg.tu-berlin.de
--               (c) 2004-2006, Institute for Cartography and Geoinformation,
--               Universität Bonn, Germany
--               http://www.ikg.uni-bonn.de
--               (c) 2005-2006, lat/lon GmbH, Germany
--               http://www.lat-lon.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0     | 2007-12-10 | release version | TKol
--                                     | LPlu
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
-- DROP TABLE "ORTHOPHOTO_RDT_IMP" CASCADE CONSTRAINT PURGE;

CREATE TABLE "ORTHOPHOTO_RDT_IMP" OF SDO_RASTER (
  PRIMARY KEY ( RASTERID, PYRAMIDLEVEL, BANDBLOCKNUMBER, ROWBLOCKNUMBER, COLUMNBLOCKNUMBER )
  LOB(RASTERBLOCK) STORE AS (NOCACHE NOLOGGING);
```

ORTHOPHOTO_RDT_IMP_ID_TRIGGER.sql

```
-- ORTHOPHOTO_RDT_IMP_ID_TRIGGER.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--               Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--               Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--               Viktor Stroh <stroh@ikg.uni-bonn.de>
--               Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2007,      Institute for Geodesy and Geoinformation Science,
--               Technische Universität Berlin, Germany
--               http://www.igg.tu-berlin.de
--               (c) 2004-2006, Institute for Cartography and Geoinformation,
--               Universität Bonn, Germany
--               http://www.ikg.uni-bonn.de
--               (c) 2005-2006, lat/lon GmbH, Germany
--               http://www.lat-lon.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0     | 2007-12-10 | release version | TKol
--                                     | LPlu
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
drop trigger ORTHO_RDT_IMP_TRG;

create trigger ORTHO_RDT_IMP_TRG
before insert on ORTHOPHOTO_RDT_IMP
for each row
begin
    if :new.id is null then
        select ORTHOPHOTO_RDT_IMP_SEQ.nextval into :new.id from dual;
    end if;
end;
/
```

ORTHOPHOTO_SEQ.sql

```
-- ORTHOPHOTO_SEQ.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--              Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--              Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--              Viktor Stroh <stroh@ikg.uni-bonn.de>
--              Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2007,      Institute for Geodesy and Geoinformation Science,
--              Technische Universität Berlin, Germany
--              http://www.igg.tu-berlin.de
--              (c) 2004-2006, Institute for Cartography and Geoinformation,
--              Universität Bonn, Germany
--              http://www.ikg.uni-bonn.de
--              (c) 2005-2006, lat/lon GmbH, Germany
--              http://www.lat-lon.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0     | 2007-12-10 | release version | TKol
--                                     | LPlu
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
-- DROP SEQUENCE "ORTHOPHOTO_IMP_SEQ";
CREATE SEQUENCE "ORTHOPHOTO_IMP_SEQ" INCREMENT BY 1 START WITH 1 MINVALUE 1;
--
-- DROP SEQUENCE "ORTHOPHOTO_RDT_SEQ";
CREATE SEQUENCE "ORTHOPHOTO_RDT_SEQ" INCREMENT BY 1 START WITH 1 MINVALUE 1;
--
-- DROP SEQUENCE "ORTHOPHOTO_RDT_IMP_SEQ";
CREATE SEQUENCE "ORTHOPHOTO_RDT_IMP_SEQ" INCREMENT BY 1 START WITH 1 MINVALUE 1;
```

TRANSPORTATION

TRAFFIC_AREA.sql

```
-- TRAFFIC_AREA.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008 Institute for Geodesy and Geoinformation Science,
--              Technische Universität Berlin, Germany
--              http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     | GKoe
--                                     | CNag
--                                     | ASta
--
CREATE TABLE TRAFFIC_AREA
(
  ID NUMBER NOT NULL,
  IS_AUXILIARY NUMBER(1),
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  FUNCTION VARCHAR2(1000),
  USAGE VARCHAR2(1000),
  SURFACE_MATERIAL VARCHAR2(256),
  LOD2_MULTI_SURFACE_ID NUMBER,
  LOD3_MULTI_SURFACE_ID NUMBER,
  LOD4_MULTI_SURFACE_ID NUMBER,
  TRANSPORTATION_COMPLEX_ID NUMBER NOT NULL
)
;
ALTER TABLE TRAFFIC_AREA
ADD CONSTRAINT TRAFFIC_AREA_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

TRANSPORTATION_COMPLEX.sql

```
-- TRANSPORTATION_COMPLEX.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE TRANSPORTATION_COMPLEX
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  FUNCTION VARCHAR2(1000),
  USAGE VARCHAR2(1000),
  TYPE VARCHAR2(256),
  LOD0_NETWORK MDSYS.SDO_GEOMETRY,
  LOD1_MULTI_SURFACE_ID NUMBER,
  LOD2_MULTI_SURFACE_ID NUMBER,
  LOD3_MULTI_SURFACE_ID NUMBER,
  LOD4_MULTI_SURFACE_ID NUMBER
)
;
ALTER TABLE TRANSPORTATION_COMPLEX
ADD CONSTRAINT TRANSPORTATION_COMPLEX_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```


VEGETATION

PLANT_COVER.sql

```
-- PLANT_COVER.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASta
--
CREATE TABLE PLANT_COVER
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  CLASS VARCHAR2(256),
  FUNCTION VARCHAR2(1000),
  AVERAGE_HEIGHT BINARY_DOUBLE,
  LOD1_GEOMETRY_ID NUMBER,
  LOD2_GEOMETRY_ID NUMBER,
  LOD3_GEOMETRY_ID NUMBER,
  LOD4_GEOMETRY_ID NUMBER
)
;
ALTER TABLE PLANT_COVER
ADD CONSTRAINT PLANT_COVER_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

SOLITARY_VEGETAT_OBJECT.sql

```
-- SOLITARY_VEGETAT_OBJECT.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     Asta
--
CREATE TABLE SOLITARY_VEGETAT_OBJECT
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  CLASS VARCHAR2(256),
  SPECIES VARCHAR2(1000),
  FUNCTION VARCHAR2(1000),
  HEIGHT BINARY DOUBLE,
  TRUNC_DIAMETER BINARY DOUBLE,
  CROWN_DIAMETER BINARY DOUBLE,
  LOD1_GEOMETRY_ID NUMBER,
  LOD2_GEOMETRY_ID NUMBER,
  LOD3_GEOMETRY_ID NUMBER,
  LOD4_GEOMETRY_ID NUMBER,
  LOD1_IMPLICIT_REP_ID NUMBER,
  LOD2_IMPLICIT_REP_ID NUMBER,
  LOD3_IMPLICIT_REP_ID NUMBER,
  LOD4_IMPLICIT_REP_ID NUMBER,
  LOD1_IMPLICIT_REF_POINT MDSYS.SDO_GEOMETRY,
  LOD2_IMPLICIT_REF_POINT MDSYS.SDO_GEOMETRY,
  LOD3_IMPLICIT_REF_POINT MDSYS.SDO_GEOMETRY,
  LOD4_IMPLICIT_REF_POINT MDSYS.SDO_GEOMETRY,
  LOD1_IMPLICIT_TRANSFORMATION VARCHAR2(1000),
  LOD2_IMPLICIT_TRANSFORMATION VARCHAR2(1000),
  LOD3_IMPLICIT_TRANSFORMATION VARCHAR2(1000),
  LOD4_IMPLICIT_TRANSFORMATION VARCHAR2(1000)
);
ALTER TABLE SOLITARY_VEGETAT_OBJECT
ADD CONSTRAINT SOLITARY_VEGETATION_OBJEC_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

WATERBODY

WATERBODY.sql

```
-- WATERBODY.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     | GKoe
--                                     | CNag
--                                     | ASta
--
CREATE TABLE WATERBODY
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  CLASS VARCHAR2(256),
  FUNCTION VARCHAR2(1000),
  USAGE VARCHAR2(1000),
  LOD0_MULTI_CURVE MDSYS.SDO_GEOMETRY,
  LOD1_MULTI_CURVE MDSYS.SDO_GEOMETRY,
  LOD1_SOLID_ID NUMBER,
  LOD2_SOLID_ID NUMBER,
  LOD3_SOLID_ID NUMBER,
  LOD4_SOLID_ID NUMBER,
  LOD0_MULTI_SURFACE_ID NUMBER,
  LOD1_MULTI_SURFACE_ID NUMBER
)
;
ALTER TABLE WATERBODY
ADD CONSTRAINT WATERBODY_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

WATERBOUNDARY_SURFACE.sql

```
-- WATERBOUNDARY_SURFACE.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE WATERBOUNDARY_SURFACE
(
  ID NUMBER NOT NULL,
  NAME VARCHAR2(1000),
  NAME_CODESPACE VARCHAR2(4000),
  DESCRIPTION VARCHAR2(4000),
  TYPE VARCHAR2(256),
  WATER_LEVEL VARCHAR2(256),
  LOD2_SURFACE_ID NUMBER,
  LOD3_SURFACE_ID NUMBER,
  LOD4_SURFACE_ID NUMBER
)
;
ALTER TABLE WATERBOUNDARY_SURFACE
ADD CONSTRAINT WATERBOUNDARY_SURFACE_PK PRIMARY KEY
(
  ID
)
ENABLE
;
```

WATERBOD_TO_WATERBND_SRF.sql

```
-- WATERBOD_TO_WATERBND_SRF.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASa
--
CREATE TABLE WATERBOD_TO_WATERBND_SRF
(
  WATERBOUNDARY_SURFACE_ID NUMBER NOT NULL,
  WATERBODY_ID NUMBER NOT NULL
)
;
ALTER TABLE WATERBOD_TO_WATERBND_SRF
ADD CONSTRAINT WATERBOD_TO_WATERBND_PK PRIMARY KEY
(
  WATERBOUNDARY_SURFACE_ID,
  WATERBODY_ID
)
ENABLE
;
```

9.2 Constraints

ADD_CONSTRAINTS.sql

```
-- ADD_CONSTRAINTS.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008 Institute for Geodesy and Geoinformation Science,
--              Technische Universität Berlin, Germany
--              http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     ASta
--
--//ADDRESS_TO_BUILDING CONSTRAINTS

ALTER TABLE ADDRESS_TO_BUILDING
ADD CONSTRAINT ADDRESS_TO_BUILDING_FK FOREIGN KEY (BUILDING_ID)
REFERENCES BUILDING (ID) ENABLE;

ALTER TABLE ADDRESS_TO_BUILDING
ADD CONSTRAINT ADDRESS_TO_BUILDING_ADDRESS_FK FOREIGN KEY (ADDRESS_ID)
REFERENCES ADDRESS (ID) ENABLE;

--//APPEARANCE CONSTRAINTS

ALTER TABLE APPEARANCE
ADD CONSTRAINT APPEARANCE_CITYMODEL_FK FOREIGN KEY (CITYMODEL_ID)
REFERENCES CITYMODEL (ID) ENABLE;

ALTER TABLE APPEARANCE
ADD CONSTRAINT APPEARANCE_CITYOBJECT_FK FOREIGN KEY (CITYOBJECT_ID)
REFERENCES CITYOBJECT (ID) ENABLE;

--//APPEAR_TO_SURFACE_DATA CONSTRAINT

ALTER TABLE APPEAR_TO_SURFACE_DATA
ADD CONSTRAINT APPEAR_TO_SURFACE_DATA_FK1 FOREIGN KEY (APPEARANCE_ID)
REFERENCES APPEARANCE (ID) ENABLE;

ALTER TABLE APPEAR_TO_SURFACE_DATA
ADD CONSTRAINT APPEAR_TO_SURFACE_DATA_FK FOREIGN KEY (SURFACE_DATA_ID)
REFERENCES SURFACE_DATA (ID) ENABLE;

--//BREAKLINE_RELIEF CONSTRAINT

ALTER TABLE BREAKLINE_RELIEF
ADD CONSTRAINT BREAKLINE_RELIEF_FK FOREIGN KEY (ID)
REFERENCES RELIEF_COMPONENT (ID) ENABLE;

--//BUILDING CONSTRAINT

ALTER TABLE BUILDING
ADD CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK FOREIGN KEY (LOD1_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE BUILDING
ADD CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK3 FOREIGN KEY (LOD4_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE BUILDING
ADD CONSTRAINT BUILDING_CITYOBJECT_FK FOREIGN KEY (ID)
```

```

REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE BUILDING
ADD CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK1 FOREIGN KEY (LOD2_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE BUILDING
ADD CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK2 FOREIGN KEY (LOD3_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE BUILDING
ADD CONSTRAINT BUILDING_BUILDING_FK FOREIGN KEY (BUILDING_PARENT_ID)
REFERENCES BUILDING (ID) ENABLE;

ALTER TABLE BUILDING
ADD CONSTRAINT BUILDING_BUILDING_FK1 FOREIGN KEY (BUILDING_ROOT_ID)
REFERENCES BUILDING (ID) ENABLE;

--//BUILDING_FURNITURE CONSTRAINT

ALTER TABLE BUILDING_FURNITURE
ADD CONSTRAINT BUILDING_FURNITURE_FK1 FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE BUILDING_FURNITURE
ADD CONSTRAINT BUILDING_FURNITURE_FK2 FOREIGN KEY (LOD4_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE BUILDING_FURNITURE
ADD CONSTRAINT BUILDING_FURNITURE_FK FOREIGN KEY (LOD4_IMPLICIT_REP_ID)
REFERENCES IMPLICIT_GEOMETRY (ID) ENABLE;

ALTER TABLE BUILDING_FURNITURE
ADD CONSTRAINT BUILDING_FURNITURE_ROOM_FK FOREIGN KEY (ROOM_ID)
REFERENCES ROOM (ID) ENABLE;

--//BUILDING_INSTALLATION CONSTRAINT

ALTER TABLE BUILDING_INSTALLATION
ADD CONSTRAINT BUILDING_INSTALLATION_FK3 FOREIGN KEY (LOD3_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE BUILDING_INSTALLATION
ADD CONSTRAINT BUILDING_INSTALLATION_FK FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE BUILDING_INSTALLATION
ADD CONSTRAINT BUILDING_INSTALLATION_ROOM_FK FOREIGN KEY (ROOM_ID)
REFERENCES ROOM (ID) ENABLE;

ALTER TABLE BUILDING_INSTALLATION
ADD CONSTRAINT BUILDING_INSTALLATION_FK4 FOREIGN KEY (LOD4_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE BUILDING_INSTALLATION
ADD CONSTRAINT BUILDING_INSTALLATION_FK1 FOREIGN KEY (BUILDING_ID)
REFERENCES BUILDING (ID) ENABLE;

ALTER TABLE BUILDING_INSTALLATION
ADD CONSTRAINT BUILDING_INSTALLATION_FK2 FOREIGN KEY (LOD2_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

--//CITYOBJECT CONSTRAINT

ALTER TABLE CITYOBJECT
ADD CONSTRAINT CITYOBJECT_OBJECTCLASS_FK FOREIGN KEY (CLASS_ID)
REFERENCES OBJECTCLASS (ID) ENABLE;

--//CITYOBJECTGROUP CONSTRAINT

ALTER TABLE CITYOBJECTGROUP
ADD CONSTRAINT CITYOBJECT_GROUP_FK FOREIGN KEY (SURFACE_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE CITYOBJECTGROUP
ADD CONSTRAINT CITYOBJECTGROUP_CITYOBJECT_FK FOREIGN KEY (PARENT_CITYOBJECT_ID)
REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE CITYOBJECTGROUP
ADD CONSTRAINT CITYOBJECTGROUP_CITYOBJECT_FK1 FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

--//CITYOBJECT_GENERICATTRIB CONSTRAINT

```

```

ALTER TABLE CITYOBJECT_GENERICATTRIB
ADD CONSTRAINT CITYOBJECT_GENERICATTRIB_FK FOREIGN KEY (CITYOBJECT_ID)
REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE CITYOBJECT_GENERICATTRIB
ADD CONSTRAINT CITYOBJECT_GENERICATTRIB_FK1 FOREIGN KEY (SURFACE_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

--//CITYOBJECT_MEMBER CONSTRAINT

ALTER TABLE CITYOBJECT_MEMBER
ADD CONSTRAINT CITYOBJECT_MEMBER_CITYMODEL_FK FOREIGN KEY (CITYMODEL_ID)
REFERENCES CITYMODEL (ID) ENABLE;

ALTER TABLE CITYOBJECT_MEMBER
ADD CONSTRAINT CITYOBJECT_MEMBER_FK FOREIGN KEY (CITYOBJECT_ID)
REFERENCES CITYOBJECT (ID) ENABLE;

--//CITY_FURNITURE CONSTRAINT

ALTER TABLE CITY_FURNITURE
ADD CONSTRAINT CITY_FURNITURE_FK FOREIGN KEY (LOD1_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE CITY_FURNITURE
ADD CONSTRAINT CITY_FURNITURE_FK1 FOREIGN KEY (LOD2_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE CITY_FURNITURE
ADD CONSTRAINT CITY_FURNITURE_FK2 FOREIGN KEY (LOD3_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE CITY_FURNITURE
ADD CONSTRAINT CITY_FURNITURE_FK3 FOREIGN KEY (LOD4_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE CITY_FURNITURE
ADD CONSTRAINT CITY_FURNITURE_FK4 FOREIGN KEY (LOD1_IMPLICIT_REP_ID)
REFERENCES IMPLICIT_GEOMETRY (ID) ENABLE;

ALTER TABLE CITY_FURNITURE
ADD CONSTRAINT CITY_FURNITURE_FK5 FOREIGN KEY (LOD2_IMPLICIT_REP_ID)
REFERENCES IMPLICIT_GEOMETRY (ID) ENABLE;

ALTER TABLE CITY_FURNITURE
ADD CONSTRAINT CITY_FURNITURE_FK6 FOREIGN KEY (LOD3_IMPLICIT_REP_ID)
REFERENCES IMPLICIT_GEOMETRY (ID) ENABLE;

ALTER TABLE CITY_FURNITURE
ADD CONSTRAINT CITY_FURNITURE_FK7 FOREIGN KEY (LOD4_IMPLICIT_REP_ID)
REFERENCES IMPLICIT_GEOMETRY (ID) ENABLE;

ALTER TABLE CITY_FURNITURE
ADD CONSTRAINT CITY_FURNITURE_CITYOBJECT_FK FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

--//EXTERNAL_REFERENCE CONSTRAINT

ALTER TABLE EXTERNAL_REFERENCE
ADD CONSTRAINT EXTERNAL_REFERENCE_FK FOREIGN KEY (CITYOBJECT_ID)
REFERENCES CITYOBJECT (ID) ENABLE;

--//GENERALIZATION CONSTRAINT

ALTER TABLE GENERALIZATION
ADD CONSTRAINT GENERALIZATION_FK1 FOREIGN KEY (GENERALIZES_TO_ID)
REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE GENERALIZATION
ADD CONSTRAINT GENERALIZATION_FK FOREIGN KEY (CITYOBJECT_ID)
REFERENCES CITYOBJECT (ID) ENABLE;

--//GENERIC_CITYOBJECT CONSTRAINT

ALTER TABLE GENERIC_CITYOBJECT
ADD CONSTRAINT GENERIC_CITYOBJECT_FK FOREIGN KEY (LOD0_IMPLICIT_REP_ID)
REFERENCES IMPLICIT_GEOMETRY (ID) ENABLE;

ALTER TABLE GENERIC_CITYOBJECT
ADD CONSTRAINT GENERIC_CITYOBJECT_FK1 FOREIGN KEY (LOD1_IMPLICIT_REP_ID)
REFERENCES IMPLICIT_GEOMETRY (ID) ENABLE;

ALTER TABLE GENERIC_CITYOBJECT
ADD CONSTRAINT GENERIC_CITYOBJECT_FK2 FOREIGN KEY (LOD2_IMPLICIT_REP_ID)

```



```

REFERENCES IMPLICIT_GEOMETRY (ID) ENABLE;

ALTER TABLE GENERIC_CITYOBJECT
ADD CONSTRAINT GENERIC_CITYOBJECT_FK3 FOREIGN KEY (LOD3_IMPLICIT_REP_ID)
REFERENCES IMPLICIT_GEOMETRY (ID) ENABLE;

ALTER TABLE GENERIC_CITYOBJECT
ADD CONSTRAINT GENERIC_CITYOBJECT_FK4 FOREIGN KEY (LOD4_IMPLICIT_REP_ID)
REFERENCES IMPLICIT_GEOMETRY (ID) ENABLE;

ALTER TABLE GENERIC_CITYOBJECT
ADD CONSTRAINT GENERIC_CITYOBJECT_FK5 FOREIGN KEY (LOD0_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE GENERIC_CITYOBJECT
ADD CONSTRAINT GENERIC_CITYOBJECT_FK6 FOREIGN KEY (LOD1_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE GENERIC_CITYOBJECT
ADD CONSTRAINT GENERIC_CITYOBJECT_FK7 FOREIGN KEY (LOD2_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE GENERIC_CITYOBJECT
ADD CONSTRAINT GENERIC_CITYOBJECT_FK8 FOREIGN KEY (LOD3_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE GENERIC_CITYOBJECT
ADD CONSTRAINT GENERIC_CITYOBJECT_FK9 FOREIGN KEY (LOD4_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE GENERIC_CITYOBJECT
ADD CONSTRAINT GENERIC_CITYOBJECT_FK10 FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

--//GROUP_TO_CITYOBJECT CONSTRAINT

ALTER TABLE GROUP_TO_CITYOBJECT
ADD CONSTRAINT GROUP_TO_CITYOBJECT_FK FOREIGN KEY (CITYOBJECT_ID)
REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE GROUP_TO_CITYOBJECT
ADD CONSTRAINT GROUP_TO_CITYOBJECT_FK1 FOREIGN KEY (CITYOBJECTGROUP_ID)
REFERENCES CITYOBJECTGROUP (ID) ENABLE;

ALTER TABLE IMPLICIT_GEOMETRY
ADD CONSTRAINT IMPLICIT_GEOMETRY_FK FOREIGN KEY (RELATIVE_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

--//LAND_USE CONSTRAINT

ALTER TABLE LAND_USE
ADD CONSTRAINT LAND_USE_CITYOBJECT_FK FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE LAND_USE
ADD CONSTRAINT LAND_USE_SURFACE_GEOMETRY_FK FOREIGN KEY (LOD0_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE LAND_USE
ADD CONSTRAINT LAND_USE_SURFACE_GEOMETRY_FK1 FOREIGN KEY (LOD1_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE LAND_USE
ADD CONSTRAINT LAND_USE_SURFACE_GEOMETRY_FK2 FOREIGN KEY (LOD2_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE LAND_USE
ADD CONSTRAINT LAND_USE_SURFACE_GEOMETRY_FK3 FOREIGN KEY (LOD3_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE LAND_USE
ADD CONSTRAINT LAND_USE_SURFACE_GEOMETRY_FK4 FOREIGN KEY (LOD4_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

--//MASSPOINT_RELIEF CONSTRAINT

ALTER TABLE MASSPOINT_RELIEF
ADD CONSTRAINT MASSPOINT_RELIEF_FK FOREIGN KEY (ID)
REFERENCES RELIEF_COMPONENT (ID) ENABLE;

--//OBJECTCLASS CONSTRAINT

ALTER TABLE OBJECTCLASS
ADD CONSTRAINT OBJECTCLASS_OBJECTCLASS_FK FOREIGN KEY (SUPERCLASS_ID)
REFERENCES OBJECTCLASS (ID) ENABLE;

```

```
--//OPENING CONSTRAINT

ALTER TABLE OPENING
ADD CONSTRAINT OPENING_SURFACE_GEOMETRY_FK1 FOREIGN KEY (LOD4_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE OPENING
ADD CONSTRAINT OPENING_CITYOBJECT_FK FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE OPENING
ADD CONSTRAINT OPENING_SURFACE_GEOMETRY_FK FOREIGN KEY (LOD3_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE OPENING
ADD CONSTRAINT OPENING_ADDRESS_FK FOREIGN KEY (ADDRESS_ID)
REFERENCES ADDRESS (ID) ENABLE;

--//OPENING_TO_THEM_SURFACE CONSTRAINT

ALTER TABLE OPENING_TO_THEM_SURFACE
ADD CONSTRAINT OPENING_TO_THEMATIC_SURFACE_FK FOREIGN KEY (OPENING_ID)
REFERENCES OPENING (ID) ENABLE;

ALTER TABLE OPENING_TO_THEM_SURFACE
ADD CONSTRAINT OPENING_TO_THEMATIC_SURFACE_FK1 FOREIGN KEY (THEMATIC_SURFACE_ID)
REFERENCES THEMATIC_SURFACE (ID) ENABLE;

--//PLANT_COVER CONSTRAINT

ALTER TABLE PLANT_COVER
ADD CONSTRAINT PLANT_COVER_FK FOREIGN KEY (LOD1_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE PLANT_COVER
ADD CONSTRAINT PLANT_COVER_FK1 FOREIGN KEY (LOD2_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE PLANT_COVER
ADD CONSTRAINT PLANT_COVER_FK3 FOREIGN KEY (LOD4_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE PLANT_COVER
ADD CONSTRAINT PLANT_COVER_FK2 FOREIGN KEY (LOD3_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE PLANT_COVER
ADD CONSTRAINT PLANT_COVER_CITYOBJECT_FK FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

--//RELIEF_COMPONENT CONSTRAINT

ALTER TABLE RELIEF_COMPONENT
ADD CONSTRAINT RELIEF_COMPONENT_CITYOBJECT_FK FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE RELIEF_COMPONENT
ADD CONSTRAINT RELIEF_COMPONENT_CHK1 CHECK (LOD>=0 AND LOD<5) ENABLE;

--//RELIEF_FEATURE CONSTRAINT

ALTER TABLE RELIEF_FEATURE
ADD CONSTRAINT RELIEF_FEATURE_CITYOBJECT_FK FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE RELIEF_FEATURE
ADD CONSTRAINT RELIEF_FEATURE_CHK1 CHECK (LOD>=0 AND LOD<5) ENABLE;

--//RELIEF_FEAT_TO_REL_COMP CONSTRAINT

ALTER TABLE RELIEF_FEAT_TO_REL_COMP
ADD CONSTRAINT RELIEF_FEAT_TO_REL_COMP_FK FOREIGN KEY (RELIEF_COMPONENT_ID)
REFERENCES RELIEF_COMPONENT (ID) ENABLE;

ALTER TABLE RELIEF_FEAT_TO_REL_COMP
ADD CONSTRAINT RELIEF_FEAT_TO_REL_COMP_FK1 FOREIGN KEY (RELIEF_FEATURE_ID)
REFERENCES RELIEF_FEATURE (ID) ENABLE;

--//ROOM CONSTRAINT

ALTER TABLE ROOM
```

```

ADD CONSTRAINT ROOM_BUILDING_FK FOREIGN KEY (BUILDING_ID)
REFERENCES BUILDING (ID) ENABLE;

ALTER TABLE ROOM
ADD CONSTRAINT ROOM_SURFACE_GEOMETRY_FK FOREIGN KEY (LOD4_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE ROOM
ADD CONSTRAINT ROOM_CITYOBJECT_FK FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

--//SOLITARY_VEGETAT_OBJECT CONSTRAINT

ALTER TABLE SOLITARY_VEGETAT_OBJECT
ADD CONSTRAINT SOLITARY_VEGETAT_OBJECT_FK FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE SOLITARY_VEGETAT_OBJECT
ADD CONSTRAINT SOLITARY_VEGETAT_OBJECT_FK1 FOREIGN KEY (LOD1_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE SOLITARY_VEGETAT_OBJECT
ADD CONSTRAINT SOLITARY_VEGETAT_OBJECT_FK2 FOREIGN KEY (LOD2_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE SOLITARY_VEGETAT_OBJECT
ADD CONSTRAINT SOLITARY_VEGETAT_OBJECT_FK3 FOREIGN KEY (LOD3_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE SOLITARY_VEGETAT_OBJECT
ADD CONSTRAINT SOLITARY_VEGETAT_OBJECT_FK4 FOREIGN KEY (LOD4_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE SOLITARY_VEGETAT_OBJECT
ADD CONSTRAINT SOLITARY_VEGETAT_OBJECT_FK5 FOREIGN KEY (LOD1_IMPLICIT_REP_ID)
REFERENCES IMPLICIT_GEOMETRY (ID) ENABLE;

ALTER TABLE SOLITARY_VEGETAT_OBJECT
ADD CONSTRAINT SOLITARY_VEGETAT_OBJECT_FK6 FOREIGN KEY (LOD2_IMPLICIT_REP_ID)
REFERENCES IMPLICIT_GEOMETRY (ID) ENABLE;

ALTER TABLE SOLITARY_VEGETAT_OBJECT
ADD CONSTRAINT SOLITARY_VEGETAT_OBJECT_FK7 FOREIGN KEY (LOD3_IMPLICIT_REP_ID)
REFERENCES IMPLICIT_GEOMETRY (ID) ENABLE;

ALTER TABLE SOLITARY_VEGETAT_OBJECT
ADD CONSTRAINT SOLITARY_VEGETAT_OBJECT_FK8 FOREIGN KEY (LOD4_IMPLICIT_REP_ID)
REFERENCES IMPLICIT_GEOMETRY (ID) ENABLE;

--//SURFACE_GEOMETRY CONSTRAINT

ALTER TABLE SURFACE_GEOMETRY
ADD CONSTRAINT SURFACE_GEOMETRY_FK FOREIGN KEY (PARENT_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE SURFACE_GEOMETRY
ADD CONSTRAINT SURFACE_GEOMETRY_FK1 FOREIGN KEY (ROOT_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

--//TEXTUREPARAM CONSTRAINT

ALTER TABLE TEXTUREPARAM
ADD CONSTRAINT TEXTUREPARAM_SURFACE_GEOM_FK FOREIGN KEY (SURFACE_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE TEXTUREPARAM
ADD CONSTRAINT TEXTUREPARAM_SURFACE_DATA_FK FOREIGN KEY (SURFACE_DATA_ID)
REFERENCES SURFACE_DATA (ID) ENABLE;

--//THEMATIC_SURFACE CONSTRAINT

ALTER TABLE THEMATIC_SURFACE
ADD CONSTRAINT THEMATIC_SURFACE_ROOM_FK FOREIGN KEY (ROOM_ID)
REFERENCES ROOM (ID) ENABLE;

ALTER TABLE THEMATIC_SURFACE
ADD CONSTRAINT THEMATIC_SURFACE_BUILDING_FK FOREIGN KEY (BUILDING_ID)
REFERENCES BUILDING (ID) ENABLE;

ALTER TABLE THEMATIC_SURFACE
ADD CONSTRAINT THEMATIC_SURFACE_FK FOREIGN KEY (LOD2_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE THEMATIC_SURFACE
ADD CONSTRAINT THEMATIC_SURFACE_CITYOBJECT_FK FOREIGN KEY (ID)

```

```

REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE THEMATIC_SURFACE
ADD CONSTRAINT THEMATIC_SURFACE_FK2 FOREIGN KEY (LOD4_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE THEMATIC_SURFACE
ADD CONSTRAINT THEMATIC_SURFACE_FK1 FOREIGN KEY (LOD3_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

--//TIN_RELIEF CONSTRAINT

ALTER TABLE TIN_RELIEF
ADD CONSTRAINT TIN_RELIEF_SURFACE_GEOMETRY_FK FOREIGN KEY (SURFACE_GEOMETRY_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE TIN_RELIEF
ADD CONSTRAINT TIN_RELIEF_RELIEF_COMPONENT_FK FOREIGN KEY (ID)
REFERENCES RELIEF_COMPONENT (ID) ENABLE;

--//TRAFFIC_AREA CONSTRAINT

ALTER TABLE TRAFFIC_AREA
ADD CONSTRAINT TRAFFIC_AREA_CITYOBJECT_FK FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE TRAFFIC_AREA
ADD CONSTRAINT TRAFFIC_AREA_FK FOREIGN KEY (LOD2_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE TRAFFIC_AREA
ADD CONSTRAINT TRAFFIC_AREA_FK1 FOREIGN KEY (LOD3_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE TRAFFIC_AREA
ADD CONSTRAINT TRAFFIC_AREA_FK2 FOREIGN KEY (LOD4_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE TRAFFIC_AREA
ADD CONSTRAINT TRAFFIC_AREA_FK3 FOREIGN KEY (TRANSPORTATION_COMPLEX_ID)
REFERENCES TRANSPORTATION_COMPLEX (ID) ENABLE;

--//TRANSPORTATION_COMPLEX CONSTRAINT

ALTER TABLE TRANSPORTATION_COMPLEX
ADD CONSTRAINT TRANSPORTATION_COMPLEX_FK FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE TRANSPORTATION_COMPLEX
ADD CONSTRAINT TRANSPORTATION_COMPLEX_FK1 FOREIGN KEY (LOD1_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE TRANSPORTATION_COMPLEX
ADD CONSTRAINT TRANSPORTATION_COMPLEX_FK2 FOREIGN KEY (LOD2_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE TRANSPORTATION_COMPLEX
ADD CONSTRAINT TRANSPORTATION_COMPLEX_FK3 FOREIGN KEY (LOD3_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE TRANSPORTATION_COMPLEX
ADD CONSTRAINT TRANSPORTATION_COMPLEX_FK4 FOREIGN KEY (LOD4_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

--//WATERBODY CONSTRAINT

ALTER TABLE WATERBODY
ADD CONSTRAINT WATERBODY_CITYOBJECT_FK FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;

ALTER TABLE WATERBODY
ADD CONSTRAINT WATERBODY_SURFACE_GEOMETRY_FK1 FOREIGN KEY (LOD2_SOLID_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE WATERBODY
ADD CONSTRAINT WATERBODY_SURFACE_GEOMETRY_FK2 FOREIGN KEY (LOD3_SOLID_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE WATERBODY
ADD CONSTRAINT WATERBODY_SURFACE_GEOMETRY_FK3 FOREIGN KEY (LOD4_SOLID_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

ALTER TABLE WATERBODY
ADD CONSTRAINT WATERBODY_SURFACE_GEOMETRY_FK4 FOREIGN KEY (LOD0_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;

```

```
ALTER TABLE WATERBODY
ADD CONSTRAINT WATERBODY_SURFACE_GEOMETRY_FK5 FOREIGN KEY (LOD1_MULTI_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;
```

```
ALTER TABLE WATERBODY
ADD CONSTRAINT WATERBODY_SURFACE_GEOMETRY_FK FOREIGN KEY (LOD1_SOLID_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;
```

```
--//WATERBOD_TO_WATERBND_SRF CONSTRAINT
```

```
ALTER TABLE WATERBOD_TO_WATERBND_SRF
ADD CONSTRAINT WATERBOD_TO_WATERBND_FK FOREIGN KEY (WATERBOUNDARY_SURFACE_ID)
REFERENCES WATERBOUNDARY_SURFACE (ID) ENABLE;
```

```
ALTER TABLE WATERBOD_TO_WATERBND_SRF
ADD CONSTRAINT WATERBOD_TO_WATERBND_FK1 FOREIGN KEY (WATERBODY_ID)
REFERENCES WATERBODY (ID) ENABLE;
```

```
ALTER TABLE WATERBOUNDARY_SURFACE
ADD CONSTRAINT WATERBOUNDARY_SRF_CITYOBJ_FK FOREIGN KEY (ID)
REFERENCES CITYOBJECT (ID) ENABLE;
```

```
ALTER TABLE WATERBOUNDARY_SURFACE
ADD CONSTRAINT WATERBOUNDARY_SURFACE_FK FOREIGN KEY (LOD2_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;
```

```
ALTER TABLE WATERBOUNDARY_SURFACE
ADD CONSTRAINT WATERBOUNDARY_SURFACE_FK1 FOREIGN KEY (LOD3_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;
```

```
ALTER TABLE WATERBOUNDARY_SURFACE
ADD CONSTRAINT WATERBOUNDARY_SURFACE_FK2 FOREIGN KEY (LOD4_SURFACE_ID)
REFERENCES SURFACE_GEOMETRY (ID) ENABLE;
```

9.3 Import procedures

IMPORT_PROCEDURES.sql

```
-- IMPORT_PROCEDURES.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--              Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--              Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--              Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--              Viktor Stroh <stroh@ikg.uni-bonn.de>
--              Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2004-2006, Institute for Cartography and Geoinformation,
--              Universität Bonn, Germany
--              http://www.ikg.uni-bonn.de
--              (c) 2005-2006, lat/lon GmbH, Germany
--              http://www.lat-lon.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.0     | 2006-04-03 | release version | LPlu
--                                     | TKol
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
-- DROP TABLE "IMPORT_PROCEDURES" CASCADE CONSTRAINT PURGE;

CREATE TABLE "IMPORT_PROCEDURES" (
  "ID" NUMBER NOT NULL,
  "NAME" VARCHAR2 (256) NOT NULL,
  "DESCRIPTION" VARCHAR2 (4000) );

ALTER TABLE "IMPORT_PROCEDURES"
ADD CONSTRAINT "IMP_PROC_PK" PRIMARY KEY ( "ID" ) ENABLE;
```

DUMMY_IMPORT.sql

```
-- DUMMY_IMPORT.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--              Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--              Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--              Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--              Viktor Stroh <stroh@ikg.uni-bonn.de>
--              Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2004-2006, Institute for Cartography and Geoinformation,
--              Universität Bonn, Germany
--              http://www.ikg.uni-bonn.de
--              (c) 2005-2006, lat/lon GmbH, Germany
--              http://www.lat-lon.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-- Dummy Importprozedur, die nichts tut.
-----
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.0     | 2006-04-03 | release version | LPlu
--                                     | TKol
--                                     | GGro
--                                     | JSch
--                                     | VStr
--                                     | APot
--
CREATE OR REPLACE PROCEDURE "DUMMY"
IS
BEGIN
DBMS_OUTPUT.PUT_LINE('DUMMY');
END;
/

DELETE FROM IMPORT_PROCEDURES WHERE NAME='DUMMY';
INSERT INTO IMPORT_PROCEDURES VALUES (2,'DUMMY',
'Dummy import procedure doing nothing.');
```

9.4 Raster Mosaic

MOSAIC.sql

```
-- //////////////////////////////////////
-- //
-- // Stored procedures written in PL/SQL for raster data management
-- //
-- // Authors: Prof. Dr. Thomas H. Kolbe, Dr. Andreas Poth
-- //
-- // Last Update: 2007-12-09

-- // Set_Orthophoto_SRID - Stored procedure for setting the SRID
-- // of the imported orthophoto tiles to the database predefined CRS.
-- // This procedure must be called after importing raster tiles
-- // using the import / export tool from lat/lon, because it leaves
-- // the SRID of the imported raster tiles empty.

CREATE OR REPLACE PROCEDURE SET_ORTHOPHOTO_SRID IS
    geor          sdo_georaster;
    orthophoto_id ORTHOPHOTO_IMP.ID%TYPE;
    srs_id        DATABASE_SRS.SRID%TYPE;

CURSOR c_orthophoto_imp IS /* select all imported orthophoto tiles */
    SELECT id, orthophotoproperty FROM ORTHOPHOTO_IMP FOR UPDATE;

BEGIN

-- // Fetch SRID for this database
SELECT srid INTO srs_id FROM DATABASE_SRS;

OPEN c_orthophoto_imp;
    LOOP
        FETCH c_orthophoto_imp INTO orthophoto_id, geor;
        exit when c_orthophoto_imp%NOTFOUND;

        sdo_geor.setModelSRID(geor, srs_id);
        UPDATE ORTHOPHOTO_IMP SET orthophotoproperty=geor
            WHERE CURRENT OF c_orthophoto_imp;
    END LOOP;
CLOSE c_orthophoto_imp;

END set_Orthophoto_SRID;
/

-- // Set_Raster_Relief_SRID - Stored procedure for setting the SRID
-- // of the imported RasterRelief tiles to the database predefined CRS.
-- // This procedure must be called after importing raster tiles
-- // using the import / export tool from lat/lon, because it leaves
-- // the SRID of the imported raster tiles empty.

CREATE OR REPLACE PROCEDURE SET_RASTER_RELIEF_SRID IS
    geor          sdo_georaster;
    raster_relief_id RASTER_RELIEF_IMP.ID%TYPE;
    srs_id        DATABASE_SRS.SRID%TYPE;

CURSOR c_raster_relief_imp IS /* select all imported raster relief tiles */
    SELECT id, rasterproperty FROM RASTER_RELIEF_IMP FOR UPDATE;

BEGIN

-- // Fetch SRID for this database
SELECT srid INTO srs_id FROM DATABASE_SRS;

OPEN c_raster_relief_imp;
    LOOP
        FETCH c_raster_relief_imp INTO raster_relief_id, geor;
        exit when c_raster_relief_imp%NOTFOUND;

        sdo_geor.setModelSRID(geor, srs_id);
        UPDATE RASTER_RELIEF_IMP SET rasterproperty=geor
            WHERE CURRENT OF c_raster_relief_imp;
    END LOOP;
CLOSE c_raster_relief_imp;

END set_Raster_Relief_SRID;
/

-- // mosaicOrthophotosInitial - Stored procedure which
-- // calls the mosaic function of Oracle GeoRaster for gathering
-- // orthophoto tiles within one large raster data object for a given LOD.
-- // The big georaster is stored in the table ORTHOPHOTO with
```



```

-- // a new ID that is generated using CITYOBJECT_SEQ. This ID
-- // is printed to stdout. In order to see the ID value, database
-- // output has to be activated in SQL*Plus before calling the
-- // procedure by the following command:
-- // SET SERVEROUTPUT on
-- //
-- // PARAMETERS:
-- //   nameVal = name of the orthophoto
-- //   typeVal = type description of the orthophoto
-- //   lineageVal = lineage (sensor, source) of the orthophoto
-- //   LoDVal = LoD of the Orthophoto
-- // Example:
-- //   Execute mosaicOrthophotosInitial('Orthophoto1','True Orthophoto 0.2m','HRSC camera
flight',2);

CREATE OR REPLACE PROCEDURE mosaicOrthophotosInitial (
  nameVal VARCHAR2, typeVal VARCHAR2, lineageVal IN VARCHAR2,
  LoDVal IN NUMBER )
AS
  gr          sdo_georaster;
  geor        sdo_georaster;
  fprnt       sdo_geometry;
  orthophoto_id CITYOBJECT.ID%TYPE;
  srs_id       DATABASE_SRS.SRID%TYPE;
  tabname      VARCHAR2(50);

BEGIN
  -- // Fetch SRID for this database
  SELECT srid INTO srs_id FROM DATABASE_SRS;

  -- // Generate new ID value for the new Orthophoto-CITYOBJECT
  SELECT CITYOBJECT_SEQ.nextval INTO orthophoto_id FROM DUAL;

  -- // Use dummy footprint because real envelope is not known before
  -- // calling mosaic.
  fprnt := mdsys.sdo_geometry (3003, srs_id, null,
                                mdsys.sdo_elem_info_array (1,1003,3),
                                mdsys.sdo_ordinate_array ( 0, 0, 0, 1, 1, 1 ) );

  -- // ***** To do: GMLID and GMLID CODESPACE should be set to sensible values
  INSERT INTO CITYOBJECT ( ID, CLASS_ID, ENVELOPE, CREATION_DATE, LINEAGE )
    VALUES ( orthophoto_id, 20, fprnt, SYSDATE, lineageVal );

  -- // set the modelSRID of all raster tiles
  SET ORTHOPHOTO_SRID;
  -- // create big raster object by mosaicking the image tiles
  DBMS_OUTPUT.PUT_LINE('Mosaicking image tiles... ');

  -- // get owner of table scheme
  SELECT USER INTO tabname FROM dual;
  tabname := tabname||'.ORTHOPHOTO_RDT';
  gr := sdo_geor.init( tabname );

  sdo_geor.mosaic( 'ORTHOPHOTO_IMP', 'ORTHOPHOTOPROPERTY', gr, null );
  -- // set the CRS of the big raster
  sdo_geor.setModelSRID(gr, srs_id);
  -- // insert big raster into table ORTHOPHOTO
  INSERT INTO ORTHOPHOTO ( id, ORTHOPHOTOPROPERTY, NAME, TYPE, DATUM, LOD )
    VALUES ( orthophoto_id, gr, nameVal, typeVal, SYSDATE, LoDVal );

  -- // update footprint
  SELECT sdo_geor.generateSpatialExtent(ORTHOPHOTOPROPERTY) into fprnt
    FROM ORTHOPHOTO
    WHERE id = orthophoto_id for update;
  -- // ***** Problem: footprint is 2D, but must be 3D for CITYOBJECT
  -- UPDATE CITYOBJECT set ENVELOPE = fprnt where ID = orthophoto_id;

  -- // create pyramid
  SELECT ORTHOPHOTOPROPERTY INTO geor from ORTHOPHOTO
    where id = orthophoto_id for update;

  -- // params can be rLevel=XXX and resampling=NN | BILINEAR | AVERAGE4 |
  -- //                                     AVERAGE16 | CUBIC
  -- // e.g. 'rLevel=4, resampling=BILINEAR'
  DBMS_OUTPUT.PUT_LINE('Generating image pyramid... ');
  sdo_geor.generatePyramid( geor, 'resampling=CUBIC');
  update ORTHOPHOTO set ORTHOPHOTOPROPERTY = geor where id = orthophoto_id;

  COMMIT;
  DBMS_OUTPUT.PUT_LINE('New Orthophoto-Cityobject generated with ID '||
    orthophoto_id);
END mosaicOrthophotosInitial;
/

-- // mosaicOrthophotosUpdate - Stored procedure for updating
-- // an existing Orthophoto. This is useful if some image tiles
-- // in ORTHOPHOTO_IMP have been replaced by updated versions.
-- // The procedure calls the mosaic function for gathering

```

```
-- // orthophoto tiles within one large raster data object
-- // which then replaces the former GeoRaster of the given
-- // Orthophoto.
-- //
-- // PARAMETERS:
-- //   idVal = ID of the Orthophoto
-- //   reason = reason for the update
-- //   updatingPerson = person who initiates this update
-- // Example:
-- //   Execute mosaicOrthophotosUpdate(8197,'Update of some tiles','Mr Smith');
```

```
CREATE OR REPLACE PROCEDURE mosaicOrthophotosUpdate(
  idVal IN NUMBER, reason IN VARCHAR2, updatingPerson IN VARCHAR2 )
AS
  gr      sdo_georaster;
  geor    sdo_georaster;
  fprnt   sdo_geometry;
  srs_id  DATABASE_SRS.SRID%TYPE;
  tabname VARCHAR2(50);

BEGIN
  -- // Fetch SRID for this database
  SELECT srid INTO srs_id FROM DATABASE_SRS;

  delete from ORTHOPHOTO_RDT where RASTERID = idVal;

  -- // set the modelSRID of all raster tiles
  SET ORTHOPHOTO_SRID;
  -- // create big raster object by mosaicking the image tiles
  DBMS_OUTPUT.PUT_LINE('Mosaicking image tiles... ');

  -- // get owner of table scheme
  SELECT USER INTO tabname FROM dual;
  tabname := tabname||'.ORTHOPHOTO_RDT';
  gr := sdo_geor.init( tabname );

  sdo_geor.mosaic( 'ORTHOPHOTO_IMP', 'ORTHOPHOTOPROPERTY', gr, null );
  -- // set the CRS of the big raster
  sdo_geor.setModelSRID(gr, srs_id);
  -- // update big raster in table ORTHOPHOTO
  UPDATE ORTHOPHOTO set ORTHOPHOTOPROPERTY = gr
    where id = idVal;

  -- // update footprint
  SELECT sdo_geor.generateSpatialExtent(ORTHOPHOTOPROPERTY) into fprnt
    FROM ORTHOPHOTO WHERE id = idVal for update;

  update CITYOBJECT set ENVELOPE = fprnt, LAST_MODIFICATION_DATE = SYSDATE,
    UPDATING_PERSON = updatingPerson, REASON_FOR_UPDATE = reason;

  -- // create pyramid
  SELECT ORTHOPHOTOPROPERTY INTO geor from ORTHOPHOTO
    where id = idVal for update;

  -- // params can be rLevel=XXX and resampling=NN | BILINEAR | AVERAGE4 |
  -- //                                     AVERAGE16 | CUBIC
  -- // e.g. 'rLevel=4, resampling=BILINEAR'
  DBMS_OUTPUT.PUT_LINE('Generating image pyramid... ');
  sdo_geor.generatePyramid( geor, 'resampling=CUBIC');
  update ORTHOPHOTO set ORTHOPHOTOPROPERTY = geor where id = idVal;

  COMMIT;
END mosaicOrthophotosUpdate;
/
```

```
-- // mosaicRasterReliefInitial - Stored procedure which
-- // calls the mosaic function of Oracle GeoRaster for gathering
-- // RasterRelief tiles within one large raster data object for a
-- // given LOD. The big georaster is stored in the table RASTER_RELIEF
-- // with a new ID that is generated using CITYOBJECT_SEQ.
-- // Furthermore, a new RELIEF tuple with a new ID from
-- // CITYOBJECT_SEQ is generated of which the new RasterRelief
-- // becomes the only member. Both IDs are printed to stdout.
-- // In order to see the ID values, database output has to be
-- // activated in SQL*Plus before calling the procedure by the
-- // following command:
-- // SET SERVEROUTPUT on
-- //
-- // PARAMETERS:
-- //   gmlIdRelief = gml:id of the ReliefFeature feature
-- //   gmlIdRaster = gml:id of the Raster feature
-- //   gmlIdCodespace = Codespace for the gml:id values
-- //   nameVal = name of the orthophoto
-- //   descVal = description of the orthophoto
-- //   lineageVal = lineage (sensor, source) of the orthophoto
-- //   LoDVal = LoD of the Orthophoto
-- // Example:
-- //   Execute mosaicRasterReliefInitial('UUID_2000abcd','UUID_2000abce','UUID',
```

```

-- //          'DTM of Berlin','0.5m Raster','Photogrammetric Processing',2);

CREATE OR REPLACE PROCEDURE mosaicRasterReliefInitial (
    gmlIdRelief VARCHAR2, gmlIdRaster VARCHAR2, gmlIdCodespace VARCHAR2,
    nameVal VARCHAR2, descVal VARCHAR2, lineageVal IN VARCHAR2,
    LoDVal IN NUMBER )
AS
    gr          sdo_georaster;
    geor        sdo_georaster;
    fprnt       sdo_geometry;
    relief_id   CITYOBJECT.ID%TYPE;
    relief_component_id CITYOBJECT.ID%TYPE;
    srs_id      DATABASE_SRS.SRID%TYPE;
    tabname     VARCHAR2(50);

BEGIN
    -- // Fetch SRID for this database
    SELECT srid INTO srs_id FROM DATABASE_SRS;

    -- // generate new ID values for the two new CITYOBJECTs (RASTER_RELIEF
    -- // and RELIEF)
    SELECT CITYOBJECT_SEQ.nextval INTO relief_component_id FROM DUAL;
    SELECT CITYOBJECT_SEQ.nextval INTO relief_id FROM DUAL;

    -- // use dummy because real envelope is not known before calling mosaic
    fprnt := mdsys.sdo_geometry (3003, srs_id, null,
                                mdsys.sdo_elem_info_array (1,1003,3),
                                mdsys.sdo_ordinate_array ( 0, 0, 0, 1, 1, 1 ) );

    -- // create ReliefFeature object
    INSERT INTO CITYOBJECT ( ID, GMLID, GMLID_CODESPACE, CLASS_ID, ENVELOPE,
                            CREATION_DATE, LINEAGE )
        VALUES ( relief_id, gmlIdRelief, gmlIdCodespace, 14, fprnt, SYSDATE, lineageVal
    );

    INSERT INTO RELIEF FEATURE ( ID, NAME, DESCRIPTION, LOD )
        VALUES ( relief_id, nameVal, descVal, LoDVal);

    -- // create RASTER object
    INSERT INTO CITYOBJECT ( ID, GMLID, GMLID_CODESPACE, CLASS_ID, ENVELOPE,
                            CREATION_DATE, LINEAGE )
        VALUES ( relief_component_id, gmlIdRaster, gmlIdCodespace, 19, fprnt,
                SYSDATE, lineageVal );
    INSERT INTO RELIEF COMPONENT ( ID, NAME, DESCRIPTION, LOD )
        VALUES ( relief_component_id, nameVal, descVal, LoDVal );

    -- // set the modelSRID of all raster tiles
    SET RASTER_RELIEF_SRID;
    -- // create big raster object by mosaicking the image tiles
    DBMS_OUTPUT.PUT_LINE('Mosaicking DTM tiles...');

    -- // get owner of table scheme
    SELECT USER INTO tabname FROM dual;
    tabname := tabname||'.RASTER_RELIEF_RDT';
    gr := sdo_geor.init( tabname );

    sdo_geor.mosaic( 'RASTER_RELIEF_IMP', 'RASTERPROPERTY', gr, null );
    -- // set the CRS of the big raster
    sdo_geor.setModelSRID(gr,srs_id);
    -- // insert big raster into table RASTER_RELIEF
    INSERT INTO RASTER_RELIEF ( ID, RASTERPROPERTY )
        VALUES ( relief_component_id, gr );

    -- // update footprint
    SELECT sdo_geor.generateSpatialExtent(RASTERPROPERTY) into fprnt
        FROM RASTER_RELIEF
        WHERE id = relief_component_id for update;

    -- // set correct envelope in all respective tables
    UPDATE RELIEF_COMPONENT set EXTENT = fprnt where ID = relief_component_id;
    -- // ***** Problem: footprint is 2D, but must be 3D for CITYOBJECT
    -- UPDATE CITYOBJECT set ENVELOPE = fprnt where ID = relief_id;
    -- UPDATE CITYOBJECT set ENVELOPE = fprnt where ID = relief_component_id;

    -- // insert association between ReliefFeature and Raster
    INSERT INTO RELIEF_FEAT_TO_REL_COMP ( RELIEF_FEATURE_ID, RELIEF_COMPONENT_ID )
        VALUES ( relief_id, relief_component_id);

    -- // create pyramid
    SELECT RASTERPROPERTY INTO geor from RASTER_RELIEF
        WHERE id = relief_component_id FOR UPDATE;

    -- // params can be rLevel=XXX and resampling=NN | BILINEAR | AVERAGE4 |
    -- //                                     AVERAGE16 | CUBIC
    -- // e.g. 'rLevel=4, resampling=BILINEAR'
    DBMS_OUTPUT.PUT_LINE('Generating raster pyramid...');
    sdo_geor.generatePyramid( geor, 'resampling=CUBIC');
    update RASTER_RELIEF set RASTERPROPERTY = geor where id = relief_component_id;

    -- // update tuples in RASTER_RELIEF_IMP to point to the generated

```

```

-- // RELIEF and RASTER RELIEF tuples
UPDATE RASTER_RELIEF_IMP set RELIEF_ID=relief_id,
                             RASTER_RELIEF_ID=relief_component_id;

COMMIT;
DBMS_OUTPUT.PUT_LINE('New Raster-Cityobject generated with ID '||
                     relief_component_id);
DBMS_OUTPUT.PUT_LINE('New ReliefFeature-Cityobject generated with ID '||
                     relief_id);
END mosaicRasterReliefInitial;
/

-- // mosaicRasterReliefUpdate - Stored procedure for updating
-- // an existing RasterRelief. This is useful if some raster tiles
-- // in RASTER_RELIEF_IMP have been replaced by updated versions.
-- // The procedure calls the mosaic function for gathering
-- // raster_relief tiles within one large raster data object
-- // which then replaces the former GeoRaster of the given
-- // ReliefObject.
-- //
-- // PARAMETERS:
-- //   idVal = ID of the Raster feature
-- //   reason = reason for the update
-- //   updatingPerson = person who initiates this update
-- // Example:
-- //   Execute mosaicRasterReliefUpdate(15233,'Update of some tiles','Mr Smith');

CREATE OR REPLACE PROCEDURE mosaicRasterReliefUpdate(
  idVal IN NUMBER, reason IN VARCHAR2, updatingPerson IN VARCHAR2 )
AS
  gr      sdo_georaster;
  geor    sdo_georaster;
  fprnt   sdo_geometry;
  srs_id  DATABASE_SRS.SRID%TYPE;
  tabname VARCHAR2(50);

BEGIN
  -- // Fetch SRID for this database
  SELECT srid INTO srs_id FROM DATABASE_SRS;

  -- // discard the old raster
  -- delete from RASTER_RELIEF_RDT where RASTER_RELIEF_ID = idVal;

  -- // set the modelSRID of all imported raster tiles
  SET_RASTER_RELIEF_SRID;

  -- // get owner of table scheme
  SELECT USER INTO tabname FROM dual;
  tabname := tabname||'.RASTER_RELIEF_RDT';
  gr := sdo_geor.init( tabname );

  DBMS_OUTPUT.PUT_LINE('Mosaicking DTM tiles...');
  sdo_geor.mosaic( 'RASTER_RELIEF_IMP', 'RASTERPROPERTY', gr, null );
  -- // set the CRS of the big raster
  sdo_geor.setModelSRID(gr,srs_id);
  -- // update big raster in table RASTER_RELIEF
  UPDATE RASTER_RELIEF set RASTERPROPERTY = gr where id = idVal;

  -- // update footprint
  SELECT sdo_geor.generateSpatialExtent(RASTERPROPERTY) into fprnt
  FROM RASTER_RELIEF WHERE id = idVal for update;

  update CITYOBJECT set ENVELOPE = fprnt, LAST_MODIFICATION_DATE = SYSDATE,
    UPDATING_PERSON = updatingPerson, REASON_FOR_UPDATE = reason
  where id=idVal;
  update CITYOBJECT set ENVELOPE = fprnt, LAST_MODIFICATION_DATE = SYSDATE,
    UPDATING_PERSON = updatingPerson, REASON_FOR_UPDATE = reason
  where id=(select RELIEF_FEATURE_ID from RELIEF_FEAT_TO_REL_COMP
    where RELIEF_COMPONENT_ID=idVal);
  update RELIEF_COMPONENT set EXTENT = fprnt where id=idVal;

  -- // create pyramid
  SELECT RASTERPROPERTY INTO geor from RASTER_RELIEF
  where id = idVal for update;

  -- // params can be rLevel=XXX and resampling=NN | BILINEAR | AVERAGE4 |
  -- //                                     AVERAGE16 | CUBIC
  -- // e.g. 'rLevel=4, resampling=BILINEAR'
  DBMS_OUTPUT.PUT_LINE('Generating raster pyramid...');
  sdo_geor.generatePyramid( geor, 'resampling=CUBIC');
  update RASTER_RELIEF set RASTERPROPERTY = geor where id = idVal;

  COMMIT;
END mosaicRasterReliefUpdate;
/

```

9.5 Trigger

TRIGGER.sql

```
-- TRIGGER.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008 Institute for Geodesy and Geoinformation Science,
--              Technische Universität Berlin, Germany
--              http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--
call sdo_geor_utl.createDMLTrigger('RASTER_RELIEF','RASTERPROPERTY');
call sdo_geor_utl.createDMLTrigger('RASTER_RELIEF_IMP','RASTERPROPERTY');

call sdo_geor_utl.createDMLTrigger('ORTHOPHOTO','ORTHOPHOTOPROPERTY');
call sdo_geor_utl.createDMLTrigger('ORTHOPHOTO_IMP','ORTHOPHOTOPROPERTY');

--@@DTM/RASTER_RELIEF_RDT_ID_TRIGGER.sql
--@@DTM/RASTER_RELIEF_RDT_IMP_ID_TRIGGER.sql

--@@ORTHOPHOTO/ORTHOPHOTO_RDT_ID_TRIGGER.sql
--@@ORTHOPHOTO/ORTHOPHOTO_RDT_IMP_ID_TRIGGER.sql
```

9.6 Indexes

BUILD_SIMPLE_INDEX.sql

```
-- BUILD_SIMPLE_INDEX.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008 Institute for Geodesy and Geoinformation Science,
--              Technische Universität Berlin, Germany
--              http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--
CREATE INDEX ADDRESS_TO_BUILDING_FKX ON ADDRESS_TO_BUILDING (BUILDING_ID);
CREATE INDEX ADDRESS_TO_BUILDING_FKX1 ON ADDRESS_TO_BUILDING (ADDRESS_ID);

CREATE INDEX APPEARANCE_FKX ON APPEARANCE (CITYMODEL_ID);
CREATE INDEX APPEARANCE_FKX1 ON APPEARANCE (CITYOBJECT_ID);
CREATE INDEX APPEARANCE_INX ON APPEARANCE (GMLID, GMLID_CODESPACE);

CREATE INDEX BUILDING_FKX ON BUILDING (LOD1_GEOMETRY_ID);
CREATE INDEX BUILDING_FKX1 ON BUILDING (LOD2_GEOMETRY_ID);
CREATE INDEX BUILDING_FKX2 ON BUILDING (LOD3_GEOMETRY_ID);
CREATE INDEX BUILDING_FKX4 ON BUILDING (BUILDING_ROOT_ID);
CREATE INDEX BUILDING_FKX5 ON BUILDING (BUILDING_PARENT_ID);
CREATE INDEX BUILDING_FKX3 ON BUILDING (LOD4_GEOMETRY_ID);

CREATE INDEX BUILDING_FURNITURE_FKX ON BUILDING_FURNITURE (LOD4_GEOMETRY_ID);
CREATE INDEX BUILDING_FURNITURE_FKX1 ON BUILDING_FURNITURE (LOD4_IMPLICIT_REP_ID);
CREATE INDEX BUILDING_FURNITURE_FKX2 ON BUILDING_FURNITURE (ROOM_ID);

CREATE INDEX BUILDING_INSTALLATION_FKX ON BUILDING_INSTALLATION (LOD3_GEOMETRY_ID);
CREATE INDEX BUILDING_INSTALLATION_FKX1 ON BUILDING_INSTALLATION (ROOM_ID);
CREATE INDEX BUILDING_INSTALLATION_FKX2 ON BUILDING_INSTALLATION (LOD4_GEOMETRY_ID);
CREATE INDEX BUILDING_INSTALLATION_FKX3 ON BUILDING_INSTALLATION (LOD2_GEOMETRY_ID);
CREATE INDEX BUILDING_INSTALLATION_FKX4 ON BUILDING_INSTALLATION (BUILDING_ID);

-- CREATE INDEX CITYMODEL_SPX ON CITYMODEL (ENVELOPE);
-- CREATE INDEX CITYOBJECT_SPX ON CITYOBJECT (ENVELOPE);

CREATE INDEX CITYOBJECT_FKX ON CITYOBJECT (CLASS_ID);
CREATE INDEX CITYOBJECT_INX ON CITYOBJECT (GMLID, GMLID_CODESPACE);

CREATE INDEX CITYOBJECTGROUP_FKX ON CITYOBJECTGROUP (SURFACE_GEOMETRY_ID);
CREATE INDEX CITYOBJECTGROUP_FKX1 ON CITYOBJECTGROUP (PARENT_CITYOBJECT_ID);

CREATE INDEX CITYOBJECT_GENERICATTRIB_FKX ON CITYOBJECT_GENERICATTRIB (CITYOBJECT_ID);
CREATE INDEX CITYOBJECT_GENERICATTRIB_FKX1 ON CITYOBJECT_GENERICATTRIB (SURFACE_GEOMETRY_ID);

-- CREATE INDEX CITYOBJECT_MEMBER_FKX ON CITYOBJECT_MEMBER (CITYMODEL_ID, CITYOBJECT_ID);

CREATE INDEX CITY_FURNITURE_FKX ON CITY_FURNITURE (LOD1_GEOMETRY_ID);
CREATE INDEX CITY_FURNITURE_FKX1 ON CITY_FURNITURE (LOD2_GEOMETRY_ID);
CREATE INDEX CITY_FURNITURE_FKX2 ON CITY_FURNITURE (LOD3_GEOMETRY_ID);
CREATE INDEX CITY_FURNITURE_FKX3 ON CITY_FURNITURE (LOD4_GEOMETRY_ID);
CREATE INDEX CITY_FURNITURE_FKX4 ON CITY_FURNITURE (LOD1_IMPLICIT_REP_ID);
CREATE INDEX CITY_FURNITURE_FKX5 ON CITY_FURNITURE (LOD2_IMPLICIT_REP_ID);
CREATE INDEX CITY_FURNITURE_FKX6 ON CITY_FURNITURE (LOD3_IMPLICIT_REP_ID);
CREATE INDEX CITY_FURNITURE_FKX7 ON CITY_FURNITURE (LOD4_IMPLICIT_REP_ID);

CREATE INDEX EXTERNAL_REFERENCE_FKX ON EXTERNAL_REFERENCE (CITYOBJECT_ID);

-- CREATE INDEX GENERALIZATION_FKX ON GENERALIZATION (CITYOBJECT_ID, GENERALIZES_TO_ID);

CREATE INDEX GENERIC_CITYOBJECT_FKX ON GENERIC_CITYOBJECT (LOD0_IMPLICIT_REP_ID);
```

```

CREATE INDEX GENERIC_CITYOBJECT_FKX1 ON GENERIC_CITYOBJECT (LOD1_IMPLICIT_REP_ID);
CREATE INDEX GENERIC_CITYOBJECT_FKX2 ON GENERIC_CITYOBJECT (LOD2_IMPLICIT_REP_ID);
CREATE INDEX GENERIC_CITYOBJECT_FKX3 ON GENERIC_CITYOBJECT (LOD3_IMPLICIT_REP_ID);
CREATE INDEX GENERIC_CITYOBJECT_FKX4 ON GENERIC_CITYOBJECT (LOD4_IMPLICIT_REP_ID);
CREATE INDEX GENERIC_CITYOBJECT_FKX5 ON GENERIC_CITYOBJECT (LOD0_GEOMETRY_ID);
CREATE INDEX GENERIC_CITYOBJECT_FKX6 ON GENERIC_CITYOBJECT (LOD1_GEOMETRY_ID);
CREATE INDEX GENERIC_CITYOBJECT_FKX7 ON GENERIC_CITYOBJECT (LOD2_GEOMETRY_ID);
CREATE INDEX GENERIC_CITYOBJECT_FKX8 ON GENERIC_CITYOBJECT (LOD3_GEOMETRY_ID);
CREATE INDEX GENERIC_CITYOBJECT_FKX9 ON GENERIC_CITYOBJECT (LOD4_GEOMETRY_ID);

-- CREATE INDEX GROUP_TO_CITYOBJECT_FKX ON GROUP_TO_CITYOBJECT (CITYOBJECT_ID,
CITYOBJECTGROUP_ID);

CREATE INDEX IMPLICIT_GEOMETRY_FKX ON IMPLICIT_GEOMETRY (RELATIVE_GEOMETRY_ID);
CREATE INDEX IMPLICIT_GEOMETRY_INX ON IMPLICIT_GEOMETRY (REFERENCE_TO_LIBRARY);

CREATE INDEX LAND_USE_FKX ON LAND_USE (LOD0_MULTI_SURFACE_ID);
CREATE INDEX LAND_USE_FKX1 ON LAND_USE (LOD1_MULTI_SURFACE_ID);
CREATE INDEX LAND_USE_FKX2 ON LAND_USE (LOD2_MULTI_SURFACE_ID);
CREATE INDEX LAND_USE_FKX3 ON LAND_USE (LOD3_MULTI_SURFACE_ID);
CREATE INDEX LAND_USE_FKX4 ON LAND_USE (LOD4_MULTI_SURFACE_ID);

CREATE INDEX OBJECTCLASS_FKX ON OBJECTCLASS (SUPERCLASS_ID);

CREATE INDEX OPENING_FKX ON OPENING (ADDRESS_ID);
CREATE INDEX OPENING_FKX1 ON OPENING (LOD3_MULTI_SURFACE_ID);
CREATE INDEX OPENING_FKX2 ON OPENING (LOD4_MULTI_SURFACE_ID);

-- CREATE INDEX OPENING_TO_THEM_SURFACE_FKX ON OPENING_TO_THEM_SURFACE (OPENING_ID,
THEMATIC_SURFACE_ID);

CREATE INDEX PLANT_COVER_FKX ON PLANT_COVER (LOD1_GEOMETRY_ID);
CREATE INDEX PLANT_COVER_FKX1 ON PLANT_COVER (LOD2_GEOMETRY_ID);
CREATE INDEX PLANT_COVER_FKX2 ON PLANT_COVER (LOD3_GEOMETRY_ID);
CREATE INDEX PLANT_COVER_FKX3 ON PLANT_COVER (LOD4_GEOMETRY_ID);

-- CREATE INDEX RELIEF_FEAT_TO_REL_COMP_FKX ON RELIEF_FEAT_TO_REL_COMP (RELIEF_COMPONENT_ID,
RELIEF_FEATURE_ID);

CREATE INDEX ROOM_FKX ON ROOM (BUILDING_ID);
CREATE INDEX ROOM_FKX1 ON ROOM (LOD4_GEOMETRY_ID);

CREATE INDEX SOLITARY_VEGETAT_OBJECT_FKX1 ON SOLITARY_VEGETAT_OBJECT (LOD1_GEOMETRY_ID);
CREATE INDEX SOLITARY_VEGETAT_OBJECT_FKX2 ON SOLITARY_VEGETAT_OBJECT (LOD2_GEOMETRY_ID);
CREATE INDEX SOLITARY_VEGETAT_OBJECT_FKX3 ON SOLITARY_VEGETAT_OBJECT (LOD3_GEOMETRY_ID);
CREATE INDEX SOLITARY_VEGETAT_OBJECT_FKX4 ON SOLITARY_VEGETAT_OBJECT (LOD4_GEOMETRY_ID);
CREATE INDEX SOLITARY_VEGETAT_OBJECT_FKX5 ON SOLITARY_VEGETAT_OBJECT (LOD1_IMPLICIT_REP_ID);
CREATE INDEX SOLITARY_VEGETAT_OBJECT_FKX6 ON SOLITARY_VEGETAT_OBJECT (LOD2_IMPLICIT_REP_ID);
CREATE INDEX SOLITARY_VEGETAT_OBJECT_FKX7 ON SOLITARY_VEGETAT_OBJECT (LOD3_IMPLICIT_REP_ID);
CREATE INDEX SOLITARY_VEGETAT_OBJECT_FKX8 ON SOLITARY_VEGETAT_OBJECT (LOD4_IMPLICIT_REP_ID);

CREATE INDEX SURFACE_DATA_INX ON SURFACE_DATA (GMLID, GMLID_CODESPACE);

CREATE INDEX SURFACE_GEOMETRY_FKX ON SURFACE_GEOMETRY (PARENT_ID);
CREATE INDEX SURFACE_GEOMETRY_FKX1 ON SURFACE_GEOMETRY (ROOT_ID);
CREATE INDEX SURFACE_GEOMETRY_INX ON SURFACE_GEOMETRY (GMLID, GMLID_CODESPACE);

CREATE INDEX TEXTUREPARAM_FKX ON TEXTUREPARAM (SURFACE_GEOMETRY_ID);
CREATE INDEX TEXTUREPARAM_FKX1 ON TEXTUREPARAM (SURFACE_DATA_ID);

CREATE INDEX THEMATIC_SURFACE_FKX ON THEMATIC_SURFACE (ROOM_ID);
CREATE INDEX THEMATIC_SURFACE_FKX1 ON THEMATIC_SURFACE (BUILDING_ID);
CREATE INDEX THEMATIC_SURFACE_FKX2 ON THEMATIC_SURFACE (LOD2_MULTI_SURFACE_ID);
CREATE INDEX THEMATIC_SURFACE_FKX3 ON THEMATIC_SURFACE (LOD3_MULTI_SURFACE_ID);
CREATE INDEX THEMATIC_SURFACE_FKX4 ON THEMATIC_SURFACE (LOD4_MULTI_SURFACE_ID);

CREATE INDEX TIN_RELIEF_FKX ON TIN_RELIEF (SURFACE_GEOMETRY_ID);

CREATE INDEX TRAFFIC_AREA_FKX ON TRAFFIC_AREA (LOD2_MULTI_SURFACE_ID);
CREATE INDEX TRAFFIC_AREA_FKX1 ON TRAFFIC_AREA (LOD3_MULTI_SURFACE_ID);
CREATE INDEX TRAFFIC_AREA_FKX2 ON TRAFFIC_AREA (LOD4_MULTI_SURFACE_ID);
CREATE INDEX TRAFFIC_AREA_FKX3 ON TRAFFIC_AREA (TRANSPORTATION_COMPLEX_ID);

CREATE INDEX TRANSPORTATION_COMPLEX_FKX1 ON TRANSPORTATION_COMPLEX (LOD1_MULTI_SURFACE_ID);
CREATE INDEX TRANSPORTATION_COMPLEX_FKX2 ON TRANSPORTATION_COMPLEX (LOD2_MULTI_SURFACE_ID);
CREATE INDEX TRANSPORTATION_COMPLEX_FKX3 ON TRANSPORTATION_COMPLEX (LOD3_MULTI_SURFACE_ID);
CREATE INDEX TRANSPORTATION_COMPLEX_FKX4 ON TRANSPORTATION_COMPLEX (LOD4_MULTI_SURFACE_ID);

CREATE INDEX WATERBODY_FKX1 ON WATERBODY (LOD2_SOLID_ID);
CREATE INDEX WATERBODY_FKX2 ON WATERBODY (LOD3_SOLID_ID);
CREATE INDEX WATERBODY_FKX3 ON WATERBODY (LOD4_SOLID_ID);
CREATE INDEX WATERBODY_FKX5 ON WATERBODY (LOD1_MULTI_SURFACE_ID);
CREATE INDEX WATERBODY_FKX4 ON WATERBODY (LOD0_MULTI_SURFACE_ID);
CREATE INDEX WATERBODY_FKX ON WATERBODY (LOD1_SOLID_ID);

-- CREATE INDEX WATERBOD_TO_WATERBND_SRF_FKX ON WATERBOD_TO_WATERBND_SRF
(WATERBOUNDARY_SURFACE_ID, WATERBODY_ID);

CREATE INDEX WATERBOUNDARY_SURFACE_FKX ON WATERBOUNDARY_SURFACE (LOD2_SURFACE_ID);

```

```
CREATE INDEX WATERBOUNDARY_SURFACE_FKX1 ON WATERBOUNDARY_SURFACE (LOD3_SURFACE_ID);  
CREATE INDEX WATERBOUNDARY_SURFACE_FKX2 ON WATERBOUNDARY_SURFACE (LOD4_SURFACE_ID);
```


SPATIAL_INDEX.sql

```
-- SPATIAL_INDEX.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
-- ChangeLog:
--
-- Version | Date       | Description                               | Author
-- 2.0.0   | 2007-11-23 | release version                           | TKol
--                                                | GKoe
--                                                | CNag
--
--// ENTRIES INTO USER_SDO_GEOM_METADATA WITH BOUNDING VOLUME WITH 10000km EXTENT FOR X,Y AND
11km FOR Z

SET SERVEROUTPUT ON
SET FEEDBACK ON

ALTER SESSION set NLS_TERRITORY='AMERICA';
ALTER SESSION set NLS_LANGUAGE='AMERICAN';

-- Fetch SRID from the the table DATABASE_SRS

VARIABLE SRID NUMBER;
BEGIN
    SELECT SRID INTO :SRID FROM DATABASE_SRS;
END;
/

-- Transfer the value from the bind variable :SRID to the substitution variable &SRSNO
column mc new_value SRSNO print
select :SRID mc from dual;

prompt Used SRID for spatial indexes: &SRSNO

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='CITYOBJECT' AND COLUMN_NAME='ENVELOPE';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('CITYOBJECT', 'ENVELOPE',
        MDSYS.SDO_DIM_ARRAY
        (
            MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
            MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
            MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

-- The following spatial index is inactive (commented out), because
-- column CITYOBJECT_GENERICATTRIB.GEOMVAL can be assigned 2D or 3D geometries.
-- If there would be created a spatial index, this would fix
-- dimensionality either to 2D or 3D.

--DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='CITYOBJECT_GENERICATTRIB' AND
COLUMN_NAME='GEOMVAL';
--INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
--VALUES ('CITYOBJECT_GENERICATTRIB', 'GEOMVAL',
--        MDSYS.SDO_DIM_ARRAY
--        (
--            MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
--            MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
--            MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='SURFACE_GEOMETRY' AND
COLUMN_NAME='GEOMETRY';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('SURFACE_GEOMETRY', 'GEOMETRY',
        MDSYS.SDO_DIM_ARRAY
        (
            MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
            MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
            MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);
```

```

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='BREAKLINE_RELIEF' AND
COLUMN_NAME='RIDGE_OR_VALLEY_LINES';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('BREAKLINE_RELIEF', 'RIDGE_OR_VALLEY_LINES',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='BREAKLINE_RELIEF' AND
COLUMN_NAME='BREAK_LINES';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('BREAKLINE_RELIEF', 'BREAK_LINES',
MDSYS.SDO_DIM_ARRAY ( MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='MASSPOINT_RELIEF' AND
COLUMN_NAME='RELIEF_POINTS';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('MASSPOINT_RELIEF', 'RELIEF_POINTS',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='ORTHOPHOTO_IMP' AND
COLUMN_NAME='FOOTPRINT';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('ORTHOPHOTO_IMP', 'FOOTPRINT',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='TIN_RELIEF' AND COLUMN_NAME='STOP_LINES';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('TIN_RELIEF', 'STOP_LINES',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='TIN_RELIEF' AND
COLUMN_NAME='BREAK_LINES';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('TIN_RELIEF', 'BREAK_LINES',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='TIN_RELIEF' AND
COLUMN_NAME='CONTROL_POINTS';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('TIN_RELIEF', 'CONTROL_POINTS',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='GENERIC_CITYOBJECT' AND
COLUMN_NAME='LOD0_TERRAIN_INTERSECTION';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('GENERIC_CITYOBJECT', 'LOD0_TERRAIN_INTERSECTION',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='GENERIC_CITYOBJECT' AND
COLUMN_NAME='LOD1_TERRAIN_INTERSECTION';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('GENERIC_CITYOBJECT', 'LOD1_TERRAIN_INTERSECTION',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='GENERIC_CITYOBJECT' AND
COLUMN_NAME='LOD2_TERRAIN_INTERSECTION';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('GENERIC_CITYOBJECT', 'LOD2_TERRAIN_INTERSECTION',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='GENERIC_CITYOBJECT' AND
COLUMN_NAME='LOD3_TERRAIN_INTERSECTION';

```

[illegible]

[illegible]

```

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='CITY_FURNITURE' AND
COLUMN_NAME='LOD1_IMPLICIT_REF_POINT';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('CITY_FURNITURE', 'LOD1_IMPLICIT_REF_POINT',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)) , &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='CITY_FURNITURE' AND
COLUMN_NAME='LOD2_IMPLICIT_REF_POINT';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('CITY_FURNITURE', 'LOD2_IMPLICIT_REF_POINT',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)) , &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='CITY_FURNITURE' AND
COLUMN_NAME='LOD3_IMPLICIT_REF_POINT';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('CITY_FURNITURE', 'LOD3_IMPLICIT_REF_POINT',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)) , &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='CITY_FURNITURE' AND
COLUMN_NAME='LOD4_IMPLICIT_REF_POINT';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('CITY_FURNITURE', 'LOD4_IMPLICIT_REF_POINT',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)) , &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='CITYMODEL' AND COLUMN_NAME='ENVELOPE';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('CITYMODEL', 'ENVELOPE',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)) , &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='CITYOBJECTGROUP' AND
COLUMN_NAME='GEOMETRY';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('CITYOBJECTGROUP', 'GEOMETRY',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)) , &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='RELIEF_COMPONENT' AND
COLUMN_NAME='EXTENT';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('RELIEF_COMPONENT', 'EXTENT',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005)) , &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='SURFACE_DATA' AND
COLUMN_NAME='GT_REFERENCE_POINT';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('SURFACE_DATA', 'GT_REFERENCE_POINT',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005)) , &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='TRANSPORTATION_COMPLEX' AND
COLUMN_NAME='LOD0_NETWORK';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('TRANSPORTATION_COMPLEX', 'LOD0_NETWORK',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)) , &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='SOLITARY_VEGETAT_OBJECT' AND
COLUMN_NAME='LOD1_IMPLICIT_REF_POINT';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('SOLITARY_VEGETAT_OBJECT', 'LOD1_IMPLICIT_REF_POINT',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)) , &SRSNO);

```

```

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='SOLITARY_VEGETAT_OBJECT' AND
COLUMN_NAME='LOD2_IMPLICIT_REF_POINT';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('SOLITARY_VEGETAT_OBJECT', 'LOD2_IMPLICIT_REF_POINT',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='SOLITARY_VEGETAT_OBJECT' AND
COLUMN_NAME='LOD3_IMPLICIT_REF_POINT';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('SOLITARY_VEGETAT_OBJECT', 'LOD3_IMPLICIT_REF_POINT',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='SOLITARY_VEGETAT_OBJECT' AND
COLUMN_NAME='LOD4_IMPLICIT_REF_POINT';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('SOLITARY_VEGETAT_OBJECT', 'LOD4_IMPLICIT_REF_POINT',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='WATERBODY' AND
COLUMN_NAME='LOD0_MULTI_CURVE';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('WATERBODY', 'LOD0_MULTI_CURVE',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME='WATERBODY' AND
COLUMN_NAME='LOD1_MULTI_CURVE';
INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO, SRID)
VALUES ('WATERBODY', 'LOD1_MULTI_CURVE',
MDSYS.SDO_DIM_ARRAY
(MDSYS.SDO_DIM_ELEMENT('X', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005),
MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO);

--// CREATE INDEX STATEMENTS

-- CITYOBJECT ENVELOPE
-- DROP INDEX CITYOBJECT_SPX;
CREATE INDEX CITYOBJECT_SPX ON CITYOBJECT(ENVELOPE) INDEXTYPE IS MDSYS.SPATIAL_INDEX;

-- inactive (see comment above about CITYOBJECT_GENERICATTRIB.GEOMVAL)
-- DROP INDEX GENERICATTRIB_SPX;
-- CREATE INDEX GENERICATTRIB_SPX ON CITYOBJECT_GENERICATTRIB(GEOMVAL) INDEXTYPE IS
MDSYS.SPATIAL_INDEX;

-- SURFACE GEOMETRY
-- DROP INDEX SURFACE_GEOM_SPX;
CREATE INDEX SURFACE_GEOM_SPX ON SURFACE_GEOMETRY(GEOMETRY) INDEXTYPE IS MDSYS.SPATIAL_INDEX;

-- BREAKLINE RIDGE OR VALLEY LINES
-- DROP INDEX BREAKLINE_RID_SPX;
CREATE INDEX BREAKLINE_RID_SPX ON BREAKLINE_RELIEF(RIDGE_OR_VALLEY_LINES) INDEXTYPE IS
MDSYS.SPATIAL_INDEX;

-- BREAKLINE BREAK LINES
-- DROP INDEX BREAKLINE_BREAK_SPX;
CREATE INDEX BREAKLINE_BREAK_SPX ON BREAKLINE_RELIEF(BREAK_LINES) INDEXTYPE IS
MDSYS.SPATIAL_INDEX;

-- MASSPOINT_RELIEF
-- DROP INDEX MASSPOINT_REL_SPX;
CREATE INDEX MASSPOINT_REL_SPX ON MASSPOINT_RELIEF(RELIEF_POINTS) INDEXTYPE IS
MDSYS.SPATIAL_INDEX;

-- ORTHOPHOTO_IMP
-- DROP INDEX ORTHOPHOTO_IMP_SPX;
CREATE INDEX ORTHOPHOTO_IMP_SPX ON ORTHOPHOTO_IMP(FOOTPRINT) INDEXTYPE IS MDSYS.SPATIAL_INDEX;

-- RELIEF_STOP_LINES
-- DROP INDEX TIN_RELF_STOP_SPX;
CREATE INDEX TIN_RELF_STOP_SPX ON TIN_RELIEF(STOP_LINES) INDEXTYPE IS MDSYS.SPATIAL_INDEX;

-- TIN_RELIEF_BREAK_LINES
-- DROP INDEX TIN_RELF_BREAK_SPX;
CREATE INDEX TIN_RELF_BREAK_SPX ON TIN_RELIEF(BREAK_LINES) INDEXTYPE IS MDSYS.SPATIAL_INDEX;

-- TIN_RELIEF_CONTROL_POINTS

```

```

-- DROP INDEX TIN_RELF_CTRLPTS_SPX;
CREATE INDEX TIN_RELF_CTRLPTS_SPX on TIN_RELIEF(CONTROL_POINTS) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- GENERIC CITYOBJECT LOD0 TERR
-- DROP INDEX GENERICCITY_LOD0TERR_SPX;
CREATE INDEX GENERICCITY_LOD0TERR_SPX on GENERIC_CITYOBJECT(LOD0_TERRAIN_INTERSECTION)
INDEXTYPE is MDSYS.SPATIAL_INDEX;

-- GENERIC CITYOBJECT LOD1 TERR
-- DROP INDEX GENERICCITY_LOD1TERR_SPX;
CREATE INDEX GENERICCITY_LOD1TERR_SPX on GENERIC_CITYOBJECT(LOD1_TERRAIN_INTERSECTION)
INDEXTYPE is MDSYS.SPATIAL_INDEX;

-- GENERIC CITYOBJECT LOD2 TERR
-- DROP INDEX GENERICCITY_LOD2TERR_SPX;
CREATE INDEX GENERICCITY_LOD2TERR_SPX on GENERIC_CITYOBJECT(LOD2_TERRAIN_INTERSECTION)
INDEXTYPE is MDSYS.SPATIAL_INDEX;

-- GENERIC CITYOBJECT LOD3 TERR
-- DROP INDEX GENERICCITY_LOD3TERR_SPX;
CREATE INDEX GENERICCITY_LOD3TERR_SPX on GENERIC_CITYOBJECT(LOD3_TERRAIN_INTERSECTION)
INDEXTYPE is MDSYS.SPATIAL_INDEX;

-- GENERIC CITYOBJECT LOD4 TERR
-- DROP INDEX GENERICCITY_LOD4TERR_SPX;
CREATE INDEX GENERICCITY_LOD4TERR_SPX on GENERIC_CITYOBJECT(LOD4_TERRAIN_INTERSECTION)
INDEXTYPE is MDSYS.SPATIAL_INDEX;

-- GENERIC CITYOBJECT LOD1_REF_POINT
-- DROP INDEX GENERICCITY_LOD1REFPNT_SPX;
CREATE INDEX GENERICCITY_LOD1REFPNT_SPX on GENERIC_CITYOBJECT(LOD1_IMPLICIT_REF_POINT)
INDEXTYPE is MDSYS.SPATIAL_INDEX;

-- GENERIC CITYOBJECT LOD2_REF_POINT
-- DROP INDEX GENERICCITY_LOD2REFPNT_SPX;
CREATE INDEX GENERICCITY_LOD2REFPNT_SPX on GENERIC_CITYOBJECT(LOD2_IMPLICIT_REF_POINT)
INDEXTYPE is MDSYS.SPATIAL_INDEX;

-- GENERIC CITYOBJECT LOD3_REF_POINT
-- DROP INDEX GENERICCITY_LOD3REFPNT_SPX;
CREATE INDEX GENERICCITY_LOD3REFPNT_SPX on GENERIC_CITYOBJECT(LOD3_IMPLICIT_REF_POINT)
INDEXTYPE is MDSYS.SPATIAL_INDEX;

-- GENERIC CITYOBJECT LOD4_REF_POINT
-- DROP INDEX GENERICCITY_LOD4REFPNT_SPX;
CREATE INDEX GENERICCITY_LOD4REFPNT_SPX on GENERIC_CITYOBJECT(LOD4_IMPLICIT_REF_POINT)
INDEXTYPE is MDSYS.SPATIAL_INDEX;

-- BUILDING LOD1 TERR
-- DROP INDEX BUILDING_LOD1TERR_SPX;
CREATE INDEX BUILDING_LOD1TERR_SPX on BUILDING(LOD1_TERRAIN_INTERSECTION) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- BUILDING LOD2 TERR
-- DROP INDEX BUILDING_LOD2TERR_SPX;
CREATE INDEX BUILDING_LOD2TERR_SPX on BUILDING(LOD2_TERRAIN_INTERSECTION) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- BUILDING LOD3 TERR
-- DROP INDEX BUILDING_LOD3TERR_SPX;
CREATE INDEX BUILDING_LOD3TERR_SPX on BUILDING(LOD3_TERRAIN_INTERSECTION) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- BUILDING LOD4 TERR
-- DROP INDEX BUILDING_LOD4TERR_SPX;
CREATE INDEX BUILDING_LOD4TERR_SPX on BUILDING(LOD4_TERRAIN_INTERSECTION) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- BUILDING LOD2 MULTI
-- DROP INDEX BUILDING_LOD2MULTI_SPX;
CREATE INDEX BUILDING_LOD2MULTI_SPX on BUILDING(LOD2_MULTI_CURVE) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- BUILDING LOD3 MULTI
-- DROP INDEX BUILDING_LOD3MULTI_SPX;
CREATE INDEX BUILDING_LOD3MULTI_SPX on BUILDING(LOD3_MULTI_CURVE) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- BUILDING LOD4 MULTI
-- DROP INDEX BUILDING_LOD4MULTI_SPX;
CREATE INDEX BUILDING_LOD4MULTI_SPX on BUILDING(LOD4_MULTI_CURVE) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- BUILDING FURNITURE LOD4_REF_POINT
-- DROP INDEX BUILDING_FURN_LOD4REFPNT_SPX;
CREATE INDEX BUILDING_FURN_LOD4REFPNT_SPX on BUILDING_FURNITURE(LOD4_IMPLICIT_REF_POINT)
INDEXTYPE is MDSYS.SPATIAL_INDEX;

```

```

-- CITY_FURNITURE_LOD1_TERR
-- DROP INDEX CITY_FURN_LOD1TERR_SPX;
CREATE INDEX CITY_FURN_LOD1TERR_SPX on CITY_FURNITURE(LOD1_TERRAIN_INTERSECTION) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- CITY_FURNITURE_LOD2_TERR
-- DROP INDEX CITY_FURN_LOD2TERR_SPX;
CREATE INDEX CITY_FURN_LOD2TERR_SPX on CITY_FURNITURE(LOD2_TERRAIN_INTERSECTION) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- CITY_FURNITURE_LOD3_TERR
-- DROP INDEX CITY_FURN_LOD3TERR_SPX;
CREATE INDEX CITY_FURN_LOD3TERR_SPX on CITY_FURNITURE(LOD3_TERRAIN_INTERSECTION) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- CITY_FURNITURE_LOD4_TERR
-- DROP INDEX CITY_FURN_LOD4TERR_SPX;
CREATE INDEX CITY_FURN_LOD4TERR_SPX on CITY_FURNITURE(LOD4_TERRAIN_INTERSECTION) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- CITY_FURNITURE_LOD1_REF_POINT
-- DROP INDEX CITY_FURN_LOD1REFPNT_SPX;
CREATE INDEX CITY_FURN_LOD1REFPNT_SPX on CITY_FURNITURE(LOD1_IMPLICIT_REF_POINT) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- CITY_FURNITURE_LOD2_REF_POINT
-- DROP INDEX CITY_FURN_LOD2REFPNT_SPX;
CREATE INDEX CITY_FURN_LOD2REFPNT_SPX on CITY_FURNITURE(LOD2_IMPLICIT_REF_POINT) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- CITY_FURNITURE_LOD3_REF_POINT
-- DROP INDEX CITY_FURN_LOD3REFPNT_SPX;
CREATE INDEX CITY_FURN_LOD3REFPNT_SPX on CITY_FURNITURE(LOD3_IMPLICIT_REF_POINT) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- CITY_FURNITURE_LOD4_REF_POINT
-- DROP INDEX CITY_FURN_LOD4REFPNT_SPX;
CREATE INDEX CITY_FURN_LOD4REFPNT_SPX on CITY_FURNITURE(LOD4_IMPLICIT_REF_POINT) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- CITYMODEL
-- DROP INDEX CITYMODEL_SPX;
CREATE INDEX CITYMODEL_SPX on CITYMODEL(ENVELOPE) INDEXTYPE is MDSYS.SPATIAL_INDEX;

-- CITYOBJECTGROUP
-- DROP INDEX CITYOBJECTGROUP_SPX;
CREATE INDEX CITYOBJECTGROUP_SPX on CITYOBJECTGROUP(GEOMETRY) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- RELIEF_COMPONENT
-- DROP INDEX RELIEF_COMPONENT_SPX;
CREATE INDEX RELIEF_COMPONENT_SPX on RELIEF_COMPONENT(EXTENT) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- SOLITARY_VEGETAT_OBJECT_LOD1_REF_POINT
-- DROP INDEX SOL_VEGETAT_OBJ_LOD1REFPNT_SPX;
CREATE INDEX SOL_VEGETAT_OBJ_LOD1REFPNT_SPX on
SOLITARY_VEGETAT_OBJECT(LOD1_IMPLICIT_REF_POINT) INDEXTYPE is MDSYS.SPATIAL_INDEX;

-- SOLITARY_VEGETAT_OBJECT_LOD2_REF_POINT
-- DROP INDEX SOL_VEGETAT_OBJ_LOD2REFPNT_SPX;
CREATE INDEX SOL_VEGETAT_OBJ_LOD2REFPNT_SPX on
SOLITARY_VEGETAT_OBJECT(LOD2_IMPLICIT_REF_POINT) INDEXTYPE is MDSYS.SPATIAL_INDEX;

-- SOLITARY_VEGETAT_OBJECT_LOD3_REF_POINT
-- DROP INDEX SOL_VEGETAT_OBJ_LOD3REFPNT_SPX;
CREATE INDEX SOL_VEGETAT_OBJ_LOD3REFPNT_SPX on
SOLITARY_VEGETAT_OBJECT(LOD3_IMPLICIT_REF_POINT) INDEXTYPE is MDSYS.SPATIAL_INDEX;

-- SOLITARY_VEGETAT_OBJECT_LOD4_REF_POINT
-- DROP INDEX SOL_VEGETAT_OBJ_LOD4REFPNT_SPX;
CREATE INDEX SOL_VEGETAT_OBJ_LOD4REFPNT_SPX on
SOLITARY_VEGETAT_OBJECT(LOD4_IMPLICIT_REF_POINT) INDEXTYPE is MDSYS.SPATIAL_INDEX;

-- SURFACE_DATA
-- DROP INDEX SURFACE_DATA_SPX;
CREATE INDEX SURFACE_DATA_SPX on SURFACE_DATA(GT_REFERENCE_POINT) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- TRANSPORTATION_COMPLEX
-- DROP INDEX TRANSPORTATION_COMPLEX_SPX;
CREATE INDEX TRANSPORTATION_COMPLEX_SPX on TRANSPORTATION_COMPLEX(LOD0_NETWORK) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

-- WATERBODY_LOD0_MULTI_CURVE
-- DROP INDEX WATERBODY_LODOMULTI_SPX;
CREATE INDEX WATERBODY_LODOMULTI_SPX on WATERBODY(LOD0_MULTI_CURVE) INDEXTYPE is
MDSYS.SPATIAL_INDEX;

```



```
-- WATERBODY_LOD1_MULTI_CURVE
-- DROP INDEX WATERBODY_LOD1MULTI_SPX;
CREATE INDEX WATERBODY_LOD1MULTI_SPX on WATERBODY (LOD1_MULTI_CURVE) INDEXTYPE is
MDSYS.SPATIAL_INDEX;
```

9.7 Database Versioning

ENABLEVERSIONING.sql

```
-- ENABLEVERSIONING.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                      Technische Universität Berlin, Germany
--                      http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
--
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description                                | Author
-- 2.0.1   | 2008-06-28 | log message included                      | TKol
-- 2.0.0   | 2007-11-23 | release version                          | TKol
--                                                GKoe
--                                                CNag
--
SELECT 'EnableVersioning procedure is working, that takes a while.' as message from DUAL;

EXECUTE
DBMS_WM.EnableVersioning('ADDRESS,ADDRESS TO BUILDING,APPEAR TO SURFACE DATA,APPEARANCE,BREAKLI
NE_RELIEF,BUILDING,BUILDING FURNITURE,BUILDING INSTALLATION,CITY FURNITURE,CITYMODEL,CITYOBJE
CT,CITYOBJECT_GENERICATTRIB,CITYOBJECT_MEMBER,CITYOBJECTGROUP,EXTERNAL_REFERENCE,GENERALIZATIO
N,GENERIC_CITYOBJECT,GROUP TO CITYOBJECT,IMPLICIT GEOMETRY,LAND USE,MASSPOINT_RELIEF,OPENING,O
PENING TO THEM SURFACE,PLANT COVER,RELIEF_COMPONENT,RELIEF_FEAT_TO_REL_COMP,RELIEF_FEATURE,ROO
M,SOLITARY_VEGETAT_OBJECT,SURFACE_DATA,SURFACE_GEOMETRY,TEXTUREPARAM,THEMATIC_SURFACE,TIN_RELI
EF,TRAFFIC_AREA,TRANSPORTATION_COMPLEX,WATERBOD_TO_WATERBND_SRF,WATERBODY,WATERBOUNDARY_SURFAC
E','VIEW_WO_OVERWRITE');
```

DISABLEVERSIONING.sql

```
-- DISABLEVERSIONING.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 2.0.0   | 2007-11-23 | release version | TKol
--                                     GKoe
--                                     CNag
--
EXECUTE
DBMS_WM.DisableVersioning('ADDRESS,ADDRESS TO BUILDING,APPEAR TO SURFACE DATA,APPEARANCE,BREAK
LINE_RELIEF,BUILDING,BUILDING FURNITURE,BUILDING INSTALLATION,CITY FURNITURE,CITYMODEL,CITYOBJ
ECT,CITYOBJECT_GENERICATTRIB,CITYOBJECT_MEMBER,CITYOBJECTGROUP,EXTERNAL_REFERENCE,GENERALIZATI
ON,GENERIC_CITYOBJECT,GROUP TO CITYOBJECT,IMPLICIT GEOMETRY,LAND USE,MASSPOINT_RELIEF,OPENING,
OPENING TO THEM SURFACE,PLANT COVER,RELIEF COMPONENT,RELIEF FEAT TO REL COMP,RELIEF FEATURE,RO
OM,SOLITARY_VEGETAT_OBJECT,SURFACE DATA,SURFACE GEOMETRY,TEXTUREPARAM,THEMATIC SURFACE,TIN_REL
IEF,TRAFFIC_AREA,TRANSPORTATION_COMPLEX,WATERBOD_TO_WATERBND_SRF,WATERBODY,WATERBOUNDARY_SURFA
CE',true,true);
```

9.8 Create Tables & Procedures of the Planningmanager

CREATE_PLANNINGMANAGER.sql

```
-- CREATE_PLANNINGMANAGER.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--              Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--              Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--              Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--              Viktor Stroh <stroh@ikg.uni-bonn.de>
--
-- Copyright:    (c) 2004-2006, Institute for Cartography and Geoinformation,
--              Universität Bonn, Germany
--              http://www.ikg.uni-bonn.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-- Aufruf der Einzelskripte zum Erstellen der notwendigen Tabellen, Indizes,
-- Sequenzen und Prozeduren.
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.0      2006-04-03   release version | LPlu
--                                     TKol
--                                     GGro
--                                     JSch
--                                     VStr
--
SELECT 'PlanningManager: Creating tables, sequences and stored procedures!' as message from
DUAL;

COMMIT;

SET SERVEROUTPUT ON;

-- database schema
@PLANNINGMANAGER/CREATE_TABLES.sql;
@PLANNINGMANAGER/CREATE_CONSTRAINTS.sql;
@PLANNINGMANAGER/CREATE_SPATIAL_INDEX.sql;

-- utility procedures
@PLANNINGMANAGER/CREATE_UTIL_PROCEDURES;

-- procedure bodies with return values (Java)
@PLANNINGMANAGER/CREATE_PLANNING_PROCEDUREBODYS.sql;
@PLANNINGMANAGER/CREATE_PLANNINGALTERNATIVE_PROCEDUREBODYS.sql;
@PLANNINGMANAGER/CREATE_CITYMODELASPECT_PROCEDUREBODYS.sql;

-- procedures for console output (SQL*Plus)
@PLANNINGMANAGER/CREATE_PLANNING_PROCEDURES.sql;
@PLANNINGMANAGER/CREATE_PLANNINGALTERNATIVE_PROCEDURES.sql;
@PLANNINGMANAGER/CREATE_CITYMODELASPECT_PROCEDURES.sql;

COMMIT;

SHOW ERRORS;

SELECT 'PlanningManager: Finished!' as message from DUAL;
```

DROP_PLANNINGMANAGER.sql

```
-- DROP_PLANNINGMANAGER.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--              Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--              Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--              Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--              Viktor Stroh <stroh@ikg.uni-bonn.de>
--
-- Copyright:    (c) 2004-2006, Institute for Cartography and Geoinformation,
--              Universität Bonn, Germany
--              http://www.ikg.uni-bonn.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-- Aufruf der Einzelskripte zum Löschen der Tabellen, Sequenzen und Prozeduren.
-----
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.0     | 2006-04-03 | release version | LPlu
--                                     | TKol
--                                     | GGro
--                                     | JSch
--                                     | VStr
--
SELECT 'PlanningManager: Dropping workspaces, tables, sequences and stored procedures' as
message from DUAL;

COMMIT;

EXECUTE DeleteAllPlanningAlternatives;
EXECUTE DeleteAllCityModelAspects;

DROP TABLE "CITY_MODEL_ASPECT_COMPONENT" CASCADE CONSTRAINT PURGE;
DROP TABLE "CITY_MODEL_ASPECT" CASCADE CONSTRAINT PURGE;
DROP TABLE "PLANNING_ALTERNATIVE" CASCADE CONSTRAINT PURGE;
DROP TABLE "PLANNING" CASCADE CONSTRAINT PURGE;

DROP SEQUENCE "CITY_MODEL_ASPECT_SEQ";
DROP SEQUENCE "PLANNING_ALTERNATIVE_SEQ";
DROP SEQUENCE "PLANNING_SEQ";

DROP PROCEDURE "ADDPLANNINGBDY";
DROP PROCEDURE "UPDATEPLANNINGBDY";
DROP PROCEDURE "DISCARDPLANNINGBDY";
DROP PROCEDURE "ACCEPTPLANNINGBDY";

DROP PROCEDURE "ADDPLANNINGALTERNATIVEBDY";
DROP PROCEDURE "UPDATEPLANNINGALTERNATIVEBDY";
DROP PROCEDURE "DISCARDPLANNINGALTERNATIVEBDY";
DROP PROCEDURE "GETDIFFBDY";
DROP PROCEDURE "GETALLDIFFBDY";
DROP PROCEDURE "GETCONFLICTSBDY";
DROP PROCEDURE "GETALLCONFLICTSBDY";
DROP PROCEDURE "REFRESHPLANNINGALTERNATIVEBDY";
DROP PROCEDURE "DELALLPLANNINGALTERNATIVESBDY";
DROP PROCEDURE "DELTERMPANNINGALTERNATIVESBDY";

DROP PROCEDURE "ADDCITYMODELASPECTBDY";
DROP PROCEDURE "DELETECITYMODELASPECTBDY";
DROP PROCEDURE "ADDPATOCMABDY";
DROP PROCEDURE "REMOVEPAFROMCMABDY";
DROP PROCEDURE "DELALLCITYMODELASPECTSBDY";

DROP PROCEDURE "ADDPLANNING";
DROP PROCEDURE "UPDATEPLANNING";
DROP PROCEDURE "DISCARDPLANNING";
DROP PROCEDURE "ACCEPTPLANNING";

DROP PROCEDURE "ADDPLANNINGALTERNATIVE";
DROP PROCEDURE "UPDATEPLANNINGALTERNATIVE";
DROP PROCEDURE "DISCARDPLANNINGALTERNATIVE";
DROP PROCEDURE "GETDIFF";
DROP PROCEDURE "GETALLDIFF";
DROP PROCEDURE "GETCONFLICTS";
DROP PROCEDURE "GETALLCONFLICTS";
DROP PROCEDURE "REFRESHPLANNINGALTERNATIVE";
DROP PROCEDURE "DELETEALLPLANNINGALTERNATIVES";
```

```
DROP PROCEDURE "DELETETERMPLANNINGALTERNATIVES";

DROP PROCEDURE "ADDCITYMODELASPECT";
DROP PROCEDURE "DELETECITYMODELASPECT";
DROP PROCEDURE "ADDPATOCMA";
DROP PROCEDURE "REMOVEPAFROMCMA";
DROP PROCEDURE "DELETEALLCITYMODELASPECTS";

COMMIT;

SHOW ERRORS;

SELECT 'PlanningManager: Finished!' as message from DUAL;
```

9.9 SYSDBA

SOLDNER_BERLIN_SRS_10G_R2.sql

```
-- SOLDNER_BERLIN_SRS_10G_R2.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--               Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--               Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--               Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--               Viktor Stroh <stroh@ikg.uni-bonn.de>
--               Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2004-2006, Institute for Cartography and Geoinformation,
--               Universität Bonn, Germany
--               http://www.ikg.uni-bonn.de
--               (c) 2005-2006, lat/lon GmbH, Germany
--               http://www.lat-lon.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
-- entry for 'SOLDNER' SRS
-- have to connect as sysdba for this operation
-----
-- ChangeLog:
--
-- Version | Date       | Description                                     | Author
-- 1.3     | 2008-04-14 | release version                                | LPlu
--                                                | TKol
--                                                | GGro
--                                                | JSch
--                                                | VStr
--                                                | APot
--
-- The following two commands are needed to ensure that
-- the new CRS is inserted in a proper way. For example,
-- if the client has a German NLS setting, the numbers will
-- be misinterpreted; in Germany the fraction separator is
-- ",", and not "."
--
ALTER SESSION set NLS_TERRITORY='AMERICA';
ALTER SESSION set NLS_LANGUAGE='AMERICAN';

DELETE FROM mdsys.sdo_cs_srs WHERE SRID=81989002;

INSERT INTO mdsys.sdo_cs_srs VALUES
('DHDN Soldner Berlin (EPSG 3068)', 81989002, 81989002, 'GROEGER', 'PROJCS ["DHDN / Soldner
Berlin",
    GEOGCS ["",
        DATUM ["", SPHEROID
            ["Bessel 1841", 6377397.155000, 299.152813],
            582.000000, 105.000000, 414.000000, -1.040000, -0.350000,
3.080000, 8.300000
        ], PRIMEM
            [ "Greenwich", 0.000000 ],
            UNIT ["Decimal Degree", 0.01745329251994330]
    ],
    PROJECTION
        ["Transverse Mercator"],
        PARAMETER ["Central_Meridian", 13.6272037],
        PARAMETER ["Latitude_Of_Origin", 52.41864828],
        PARAMETER ["False_Easting", 40000.00],
        PARAMETER ["False_Northing", 10000.00],
        UNIT ["Meter", 1.00000000000000]
    ], NULL);
```

9.10 Utilities

GEODB_REPORT.sql

```
-- GEODB_REPORT.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--              Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--              Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--              Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--              Viktor Stroh <stroh@ikg.uni-bonn.de>
--              Dr. Andreas Poth <poth@lat-lon.de>
--
-- Copyright:    (c) 2004-2006, Institute for Cartography and Geoinformation,
--              Universität Bonn, Germany
--              http://www.ikg.uni-bonn.de
--              (c) 2005-2006, lat/lon GmbH, Germany
--              http://www.lat-lon.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
-----
--
-- ChangeLog:
--
-- Version | Date       | Description                                     | Author
-- 1.0     | 2006-04-03   | release version                               | LPlu
--                                              | TKol
--                                              | GGro
--                                              | JSch
--                                              | VStr
--                                              | APot
--
CREATE OR REPLACE PROCEDURE GEODB_REPORT IS
-- lokale Variablen
ws VARCHAR2(30);
cnt NUMBER;
refreshDate DATE;
reportDate DATE;
pa_id PLANNING_ALTERNATIVE.ID%TYPE;
pa_title PLANNING_ALTERNATIVE.TITLE%TYPE;

BEGIN
  DBMS_OUTPUT.ENABLE;
  DBMS_OUTPUT.PUT(CHR(10));
  SELECT SYSDATE INTO reportDate FROM DUAL;
  DBMS_OUTPUT.PUT('Database Report on 3D City Model - Report date: ');
  DBMS_OUTPUT.PUT_LINE(TO_CHAR(reportDate, 'DD.MM.YYYY HH24:MI:SS'));
  DBMS_OUTPUT.PUT_LINE('=====');
  DBMS_OUTPUT.PUT(CHR(10));

  -- Determine current workspace
  ws := DBMS_WM.GetWorkspace;
  DBMS_OUTPUT.PUT('Current workspace: ' || ws);

  IF ws != 'LIVE' THEN
    -- Get associated planning alternative
    SELECT id,title INTO pa_id,pa_title FROM PLANNING_ALTERNATIVE
    WHERE workspace_name=ws;
    DBMS_OUTPUT.PUT_LINE(' (PlanningAlternative ID ' || pa_id || ': "' || pa_title || '"');

    -- Query date of last refresh
    SELECT createtime INTO refreshDate
    FROM all_workspace_savepoints
    WHERE savepoint='refreshed' AND workspace=ws;
    DBMS_OUTPUT.PUT_LINE('Last refresh from LIVE workspace: ' || TO_CHAR(refreshDate,
    'DD.MM.YYYY HH24:MI:SS'));
  END IF;
  DBMS_OUTPUT.NEW_LINE;
  DBMS_OUTPUT.PUT(CHR(10));

  SELECT count(*) INTO cnt FROM citymodel;
  DBMS_OUTPUT.PUT_LINE('#CITYMODEL: ' || SUBSTR(' ' || cnt,-8));
  SELECT count(*) INTO cnt FROM cityobject;
  DBMS_OUTPUT.PUT_LINE('#CITYOBJECT: ' || SUBSTR(' ' || cnt,-8));
  SELECT count(*) INTO cnt FROM cityobject_genericattrib;
  DBMS_OUTPUT.PUT_LINE('#CITYOBJECT_GENERICATTRIB: ' || SUBSTR(' ' || cnt,-8));
  SELECT count(*) INTO cnt FROM cityobject_member;
  DBMS_OUTPUT.PUT_LINE('#CITYOBJECT_MEMBER: ' || SUBSTR(' ' || cnt,-8));
  SELECT count(*) INTO cnt FROM city_furniture;
```



```

DBMS_OUTPUT.PUT_LINE('#CITY FURNITURE: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM external_reference;
DBMS_OUTPUT.PUT_LINE('#EXTERNAL_REFERENCE: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM generalization;
DBMS_OUTPUT.PUT_LINE('#GENERALIZATION: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM generic_cityobject;
DBMS_OUTPUT.PUT_LINE('#GENERIC_CITYOBJECT: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM implicit_geometry;
DBMS_OUTPUT.PUT_LINE('#IMPLICIT_GEOMETRY: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM objectclass;
DBMS_OUTPUT.PUT_LINE('#OBJECTCLASS: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM surface_geometry;
DBMS_OUTPUT.PUT_LINE('#SURFACE_GEOMETRY: ' || SUBSTR(' ' || cnt,-8));

SELECT count(*) INTO cnt FROM cityobjectgroup;
DBMS_OUTPUT.PUT_LINE('#CITYOBJECTGROUP: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM group_to_cityobject;
DBMS_OUTPUT.PUT_LINE('#GROUP_TO_CITYOBJECT: ' || SUBSTR(' ' || cnt,-8));

SELECT count(*) INTO cnt FROM address;
DBMS_OUTPUT.PUT_LINE('#ADDRESS: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM address_to_building;
DBMS_OUTPUT.PUT_LINE('#ADDRESS_TO_BUILDING: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM building;
DBMS_OUTPUT.PUT_LINE('#BUILDING: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM building_furniture;
DBMS_OUTPUT.PUT_LINE('#BUILDING_FURNITURE: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM building_installation;
DBMS_OUTPUT.PUT_LINE('#BUILDING_INSTALLATION: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM opening;
DBMS_OUTPUT.PUT_LINE('#OPENING: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM opening_to_them_surface;
DBMS_OUTPUT.PUT_LINE('#OPENING_TO_THEM_SURFACE: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM room;
DBMS_OUTPUT.PUT_LINE('#ROOM: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM thematic_surface;
DBMS_OUTPUT.PUT_LINE('#THEMATIC_SURFACE: ' || SUBSTR(' ' || cnt,-8));

SELECT count(*) INTO cnt FROM appearance;
DBMS_OUTPUT.PUT_LINE('#APPEARANCE: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM surface_data;
DBMS_OUTPUT.PUT_LINE('#SURFACE_DATA: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM textureparam;
DBMS_OUTPUT.PUT_LINE('#TEXTUREPARAM: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM appear_to_surface_data;
DBMS_OUTPUT.PUT_LINE('#APPEAR_TO_SURFACE_DATA: ' || SUBSTR(' ' || cnt,-8));

SELECT count(*) INTO cnt FROM breakline_relief;
DBMS_OUTPUT.PUT_LINE('#BREAKLINE_RELIEF: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM masspoint_relief;
DBMS_OUTPUT.PUT_LINE('#MASSPOINT_RELIEF: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM raster_relief;
DBMS_OUTPUT.PUT_LINE('#RASTER_RELIEF: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM raster_relief_imp;
DBMS_OUTPUT.PUT_LINE('#RASTER_RELIEF_IMP: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM relief;
DBMS_OUTPUT.PUT_LINE('#RELIEF: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM relief_component;
DBMS_OUTPUT.PUT_LINE('#RELIEF_COMPONENT: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM relief_feature;
DBMS_OUTPUT.PUT_LINE('#RELIEF_FEATURE: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM relief;
DBMS_OUTPUT.PUT_LINE('#RELIEF_FEAT_TO_REL_COMP: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM tin_relief;
DBMS_OUTPUT.PUT_LINE('#TIN_RELIEF: ' || SUBSTR(' ' || cnt,-8));

SELECT count(*) INTO cnt FROM orthophoto;
DBMS_OUTPUT.PUT_LINE('#ORTHOPHOTO: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM orthophoto_imp;
DBMS_OUTPUT.PUT_LINE('#ORTHOPHOTO_IMP: ' || SUBSTR(' ' || cnt,-8));

SELECT count(*) INTO cnt FROM transportation_complex;
DBMS_OUTPUT.PUT_LINE('#TRANSPORTATION_COMPLEX: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM traffic_area;
DBMS_OUTPUT.PUT_LINE('#TRAFFIC_AREA: ' || SUBSTR(' ' || cnt,-8));

SELECT count(*) INTO cnt FROM land_use;
DBMS_OUTPUT.PUT_LINE('#LAND_USE: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM plant_cover;
DBMS_OUTPUT.PUT_LINE('#PLANT_COVER: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM solitary_vegetat_object;
DBMS_OUTPUT.PUT_LINE('#SOLITARY_VEGETAT_OBJECT: ' || SUBSTR(' ' || cnt,-8));

SELECT count(*) INTO cnt FROM waterbody;
DBMS_OUTPUT.PUT_LINE('#WATERBODY: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM waterboundary_surface;
DBMS_OUTPUT.PUT_LINE('#WATERBOUNDARY_SURFACE: ' || SUBSTR(' ' || cnt,-8));
SELECT count(*) INTO cnt FROM waterbod_to_waterbnd_srf;
DBMS_OUTPUT.PUT_LINE('#WATERBOD_TO_WATERBND_SRF: ' || SUBSTR(' ' || cnt,-8));

```

```
SELECT count(*) INTO cnt FROM cityobject;  
DBMS_OUTPUT.PUT_LINE('#CITYOBJECT: ' || SUBSTR(' ' || cnt,-8));  
END GeoDB_Report;  
/
```

DELETE_BUILDINGS.sql

```
-- DELETE_BUILDINGS.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--               Gerhard König <gerhard.koenig@tu-berlin.de>
--               Claus Nagel <nagel@igg.tu-berlin.de>
--               Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008, Institute for Geodesy and Geoinformation Science,
--               Technische Universität Berlin, Germany
--               http://www.igg.tu-berlin.de
--
--               This script is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
--
-----
-- About: This script deletes all building instances from the table
-- BUILDING within the current workspace. This also affects all related
-- tables within the database schema.
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.0.0   | 2007-12-20 | release version | TKol
--                                     GKoe
--                                     CNag
--                                     Asta
--
--// disable FK-constraints on SURFACE_GEOMETRY
ALTER TABLE SURFACE_GEOMETRY
  DISABLE
  CONSTRAINT SURFACE_GEOMETRY_FK
  DISABLE
  CONSTRAINT SURFACE_GEOMETRY_FK1;

--// *****
--// delete openings
--// *****
SELECT 'Deleting openings...' as message from DUAL;

ALTER TABLE OPENING
  DISABLE
  CONSTRAINT OPENING_SURFACE_GEOMETRY_FK
  DISABLE
  CONSTRAINT OPENING_SURFACE_GEOMETRY_FK1;

--// delete entries from OPENING_TO_THEM_SURFACE
DELETE FROM OPENING_TO_THEM_SURFACE;

--// delete entries from TEXTUREPARAM
DELETE FROM TEXTUREPARAM
  WHERE SURFACE_GEOMETRY_ID IN
    (SELECT ID FROM SURFACE_GEOMETRY
     WHERE ROOT_ID IN
       (SELECT LOD3_MULTI_SURFACE_ID FROM OPENING));

DELETE FROM TEXTUREPARAM
  WHERE SURFACE_GEOMETRY_ID IN
    (SELECT ID FROM SURFACE_GEOMETRY
     WHERE ROOT_ID IN
       (SELECT LOD4_MULTI_SURFACE_ID FROM OPENING));

--// delete entries from SURFACE_GEOMETRY
DELETE FROM SURFACE_GEOMETRY
  WHERE ROOT_ID IN
    (SELECT LOD3_MULTI_SURFACE_ID FROM OPENING);

DELETE FROM SURFACE_GEOMETRY
  WHERE ROOT_ID IN
    (SELECT LOD4_MULTI_SURFACE_ID FROM OPENING);

--// delete entries from OPENING
DELETE FROM OPENING;

ALTER TABLE OPENING
  ENABLE
  CONSTRAINT OPENING_SURFACE_GEOMETRY_FK
  ENABLE
  CONSTRAINT OPENING_SURFACE_GEOMETRY_FK1;

--// *****
--// delete addresses
--// *****
```

```

SELECT 'Deleting addresses...' as message from DUAL;

--// delete entries from ADDRESS_TO_BUILDING;
DELETE FROM ADDRESS_TO_BUILDING;

--// delete entries from ADDRESS
DELETE FROM ADDRESS;

--// *****
--// delete thematic surfaces
--// *****
SELECT 'Deleting thematic surfaces...' as message from DUAL;

ALTER TABLE THEMATIC_SURFACE
  DISABLE
  CONSTRAINT THEMATIC_SURFACE_FK
  DISABLE
  CONSTRAINT THEMATIC_SURFACE_FK1
  DISABLE
  CONSTRAINT THEMATIC_SURFACE_FK2;

--// delete entries from TEXTUREPARAM
DELETE FROM TEXTUREPARAM
  WHERE SURFACE_GEOMETRY_ID IN
    (SELECT ID FROM SURFACE_GEOMETRY
     WHERE ROOT_ID IN
       (SELECT LOD2_MULTI_SURFACE_ID FROM THEMATIC_SURFACE));

DELETE FROM TEXTUREPARAM
  WHERE SURFACE_GEOMETRY_ID IN
    (SELECT ID FROM SURFACE_GEOMETRY
     WHERE ROOT_ID IN
       (SELECT LOD3_MULTI_SURFACE_ID FROM THEMATIC_SURFACE));

DELETE FROM TEXTUREPARAM
  WHERE SURFACE_GEOMETRY_ID IN
    (SELECT ID FROM SURFACE_GEOMETRY
     WHERE ROOT_ID IN
       (SELECT LOD4_MULTI_SURFACE_ID FROM THEMATIC_SURFACE));

--// delete entries from SURFACE_GEOMETRY
DELETE FROM SURFACE_GEOMETRY
  WHERE ROOT_ID IN
    (SELECT LOD2_MULTI_SURFACE_ID FROM THEMATIC_SURFACE);

DELETE FROM SURFACE_GEOMETRY
  WHERE ROOT_ID IN
    (SELECT LOD3_MULTI_SURFACE_ID FROM THEMATIC_SURFACE);

DELETE FROM SURFACE_GEOMETRY
  WHERE ROOT_ID IN
    (SELECT LOD4_MULTI_SURFACE_ID FROM THEMATIC_SURFACE);

--// delete entries from THEMATIC_SURFACE
DELETE FROM THEMATIC_SURFACE;

ALTER TABLE THEMATIC_SURFACE
  ENABLE
  CONSTRAINT THEMATIC_SURFACE_FK
  ENABLE
  CONSTRAINT THEMATIC_SURFACE_FK1
  ENABLE
  CONSTRAINT THEMATIC_SURFACE_FK2;

--// *****
--// delete building installations
--// *****
SELECT 'Deleting building installations...' as message from DUAL;

ALTER TABLE BUILDING_INSTALLATION
  DISABLE
  CONSTRAINT BUILDING_INSTALLATION_FK2
  DISABLE
  CONSTRAINT BUILDING_INSTALLATION_FK3
  DISABLE
  CONSTRAINT BUILDING_INSTALLATION_FK4;

--// delete entries from TEXTUREPARAM
DELETE FROM TEXTUREPARAM
  WHERE SURFACE_GEOMETRY_ID IN
    (SELECT ID FROM SURFACE_GEOMETRY
     WHERE ROOT_ID IN
       (SELECT LOD2_GEOMETRY_ID FROM BUILDING_INSTALLATION));

DELETE FROM TEXTUREPARAM
  WHERE SURFACE_GEOMETRY_ID IN
    (SELECT ID FROM SURFACE_GEOMETRY
     WHERE ROOT_ID IN

```

```

        (SELECT LOD3_GEOMETRY_ID FROM BUILDING_INSTALLATION));

DELETE FROM TEXTUREPARAM
  WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
          WHERE ROOT_ID IN
        (SELECT LOD4_GEOMETRY_ID FROM BUILDING_INSTALLATION));

--// delete entries from SURFACE_GEOMETRY
DELETE FROM SURFACE_GEOMETRY
  WHERE ROOT_ID IN
        (SELECT LOD2_GEOMETRY_ID FROM BUILDING_INSTALLATION);

DELETE FROM SURFACE_GEOMETRY
  WHERE ROOT_ID IN
        (SELECT LOD3_GEOMETRY_ID FROM BUILDING_INSTALLATION);

DELETE FROM SURFACE_GEOMETRY
  WHERE ROOT_ID IN
        (SELECT LOD4_GEOMETRY_ID FROM BUILDING_INSTALLATION);

--// delete entries from BUILDING_INSTALLATION
DELETE FROM BUILDING_INSTALLATION;

ALTER TABLE BUILDING_INSTALLATION
  ENABLE
  CONSTRAINT BUILDING_INSTALLATION_FK2
  ENABLE
  CONSTRAINT BUILDING_INSTALLATION_FK3
  ENABLE
  CONSTRAINT BUILDING_INSTALLATION_FK4;

--// *****
--// delete building furniture
--// *****
SELECT 'Deleting building furniture...' as message from DUAL;

ALTER TABLE IMPLICIT_GEOMETRY
  DISABLE
  CONSTRAINT IMPLICIT_GEOMETRY_FK;

--// delete entries from TEXTUREPARAM
DELETE FROM TEXTUREPARAM
  WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
          WHERE ROOT_ID IN
        (SELECT RELATIVE_GEOMETRY_ID FROM IMPLICIT_GEOMETRY
          WHERE ID IN
        (SELECT LOD4_IMPLICIT_REP_ID FROM
BUILDING_FURNITURE)));

--// delete entries from SURFACE_GEOMETRY
DELETE FROM SURFACE_GEOMETRY
  WHERE ROOT_ID IN
        (SELECT RELATIVE_GEOMETRY_ID FROM IMPLICIT_GEOMETRY
          WHERE ID IN
        (SELECT LOD4_IMPLICIT_REP_ID FROM BUILDING_FURNITURE));

ALTER TABLE BUILDING_FURNITURE
  DISABLE
  CONSTRAINT BUILDING_FURNITURE_FK
  DISABLE
  CONSTRAINT BUILDING_FURNITURE_FK2;

--// delete entries from IMPLICIT_GEOMETRY
DELETE FROM IMPLICIT_GEOMETRY
  WHERE ID IN
        (SELECT LOD4_IMPLICIT_REP_ID FROM BUILDING_FURNITURE);

--// delete entries from TEXTUREPARAM
DELETE FROM TEXTUREPARAM
  WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
          WHERE ROOT_ID IN
        (SELECT LOD4_GEOMETRY_ID FROM BUILDING_FURNITURE));

--// delete entries from SURFACE_GEOMETRY
DELETE FROM SURFACE_GEOMETRY
  WHERE ROOT_ID IN
        (SELECT LOD4_GEOMETRY_ID FROM BUILDING_FURNITURE);

--// delete entries from BUILDING_FURNITURE
DELETE FROM BUILDING_FURNITURE;

ALTER TABLE BUILDING_FURNITURE
  ENABLE
  CONSTRAINT BUILDING_FURNITURE_FK
  ENABLE
  CONSTRAINT BUILDING_FURNITURE_FK2;

```

```

ALTER TABLE IMPLICIT_GEOMETRY
    ENABLE
    CONSTRAINT IMPLICIT_GEOMETRY_FK;

--// *****
--// delete rooms
--// *****
SELECT 'Deleting rooms...' as message from DUAL;

ALTER TABLE ROOM
    DISABLE
    CONSTRAINT ROOM_SURFACE_GEOMETRY_FK;

--// delete entries from TEXTUREPARAM
DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
         WHERE ROOT_ID IN
             (SELECT LOD4_GEOMETRY_ID FROM ROOM));

--// delete entries from SURFACE_GEOMETRY
DELETE FROM SURFACE_GEOMETRY
    WHERE ROOT_ID IN
        (SELECT LOD4_GEOMETRY_ID FROM ROOM);

--// delete entries from ROOM
DELETE FROM ROOM;

ALTER TABLE ROOM
    ENABLE
    CONSTRAINT ROOM_SURFACE_GEOMETRY_FK;

--// *****
--// delete buildings
--// *****
SELECT 'Deleting buildings...' as message from DUAL;

ALTER TABLE BUILDING
    DISABLE
    CONSTRAINT BUILDING_BUILDING_FK
    DISABLE
    CONSTRAINT BUILDING_BUILDING_FK1
    DISABLE
    CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK
    DISABLE
    CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK1
    DISABLE
    CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK2
    DISABLE
    CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK3;

--// delete entries from TEXTUREPARAM
DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
         WHERE ROOT_ID IN
             (SELECT LOD1_GEOMETRY_ID FROM BUILDING));

DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
         WHERE ROOT_ID IN
             (SELECT LOD2_GEOMETRY_ID FROM BUILDING));

DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
         WHERE ROOT_ID IN
             (SELECT LOD3_GEOMETRY_ID FROM BUILDING));

DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
         WHERE ROOT_ID IN
             (SELECT LOD4_GEOMETRY_ID FROM BUILDING));

--// delete entries from SURFACE_GEOMETRY
DELETE FROM SURFACE_GEOMETRY
    WHERE ROOT_ID IN
        (SELECT LOD1_GEOMETRY_ID FROM BUILDING);

DELETE FROM SURFACE_GEOMETRY
    WHERE ROOT_ID IN
        (SELECT LOD2_GEOMETRY_ID FROM BUILDING);

DELETE FROM SURFACE_GEOMETRY
    WHERE ROOT_ID IN
        (SELECT LOD3_GEOMETRY_ID FROM BUILDING);

```

```

DELETE FROM SURFACE_GEOMETRY
    WHERE ROOT_ID IN
        (SELECT LOD4_GEOMETRY_ID FROM BUILDING);

--// delete entries from BUILDING
DELETE FROM BUILDING;

ALTER TABLE BUILDING
    ENABLE
    CONSTRAINT BUILDING_BUILDING_FK
    ENABLE
    CONSTRAINT BUILDING_BUILDING_FK1
    ENABLE
    CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK
    ENABLE
    CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK1
    ENABLE
    CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK2
    ENABLE
    CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK3;

--// *****
--// delete external_references
--// *****
SELECT 'Deleting external references...' as message from DUAL;

--// delete entries from EXTERNAL_REFERENCE
DELETE FROM EXTERNAL_REFERENCE
    WHERE CITYOBJECT_ID IN
        (SELECT ID FROM CITYOBJECT
            WHERE CLASS_ID=24
            OR CLASS_ID=25
            OR CLASS_ID=26
            OR CLASS_ID=27
            OR CLASS_ID=28
            OR CLASS_ID=29
            OR CLASS_ID=30
            OR CLASS_ID=31
            OR CLASS_ID=32
            OR CLASS_ID=33
            OR CLASS_ID=34
            OR CLASS_ID=35
            OR CLASS_ID=36
            OR CLASS_ID=37
            OR CLASS_ID=38
            OR CLASS_ID=39
            OR CLASS_ID=40
            OR CLASS_ID=41
            OR CLASS_ID=58);

--// *****
--// delete generic attributes
--// *****
SELECT 'Deleting generic attributes...' as message from DUAL;

ALTER TABLE CITYOBJECT_GENERICATTRIB
    DISABLE
    CONSTRAINT CITYOBJECT_GENERICATTRIB_FK1;

--// delete entries from TEXTUREPARAM
DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
            WHERE ROOT_ID IN
                (SELECT SURFACE_GEOMETRY_ID FROM CITYOBJECT_GENERICATTRIB
                    WHERE CITYOBJECT_ID IN
                        (SELECT ID FROM CITYOBJECT
                            WHERE CLASS_ID=24
                            OR CLASS_ID=25
                            OR CLASS_ID=26
                            OR CLASS_ID=27
                            OR CLASS_ID=28
                            OR CLASS_ID=29
                            OR CLASS_ID=30
                            OR CLASS_ID=31
                            OR CLASS_ID=32
                            OR CLASS_ID=33
                            OR CLASS_ID=34
                            OR CLASS_ID=35
                            OR CLASS_ID=36
                            OR CLASS_ID=37
                            OR CLASS_ID=38
                            OR CLASS_ID=39
                            OR CLASS_ID=40
                            OR CLASS_ID=41
                            OR CLASS_ID=58)));

--// delete entries from SURFACE_GEOMETRY

```

```

DELETE FROM SURFACE_GEOMETRY
  WHERE ROOT_ID IN
    (SELECT SURFACE_GEOMETRY_ID FROM CITYOBJECT_GENERICATTRIB
      WHERE CITYOBJECT_ID IN
        (SELECT ID FROM CITYOBJECT
          WHERE CLASS_ID=24
            OR CLASS_ID=25
            OR CLASS_ID=26
            OR CLASS_ID=27
            OR CLASS_ID=28
            OR CLASS_ID=29
            OR CLASS_ID=30
            OR CLASS_ID=31
            OR CLASS_ID=32
            OR CLASS_ID=33
            OR CLASS_ID=34
            OR CLASS_ID=35
            OR CLASS_ID=36
            OR CLASS_ID=37
            OR CLASS_ID=38
            OR CLASS_ID=39
            OR CLASS_ID=40
            OR CLASS_ID=41
            OR CLASS_ID=58));

--// delete entries from CITYOBJECT_GENERICATTRIB
DELETE FROM CITYOBJECT_GENERICATTRIB
  WHERE CITYOBJECT_ID IN
    (SELECT ID FROM CITYOBJECT
      WHERE CLASS_ID=24
        OR CLASS_ID=25
        OR CLASS_ID=26
        OR CLASS_ID=27
        OR CLASS_ID=28
        OR CLASS_ID=29
        OR CLASS_ID=30
        OR CLASS_ID=31
        OR CLASS_ID=32
        OR CLASS_ID=33
        OR CLASS_ID=34
        OR CLASS_ID=35
        OR CLASS_ID=36
        OR CLASS_ID=37
        OR CLASS_ID=38
        OR CLASS_ID=39
        OR CLASS_ID=40
        OR CLASS_ID=41
        OR CLASS_ID=58);

ALTER TABLE CITYOBJECT_GENERICATTRIB
  ENABLE
  CONSTRAINT CITYOBJECT_GENERICATTRIB_FK1;

--// *****
--// delete generalizations
--// *****
SELECT 'Deleting generalization hierarchies...' as message from DUAL;

--// delete entries from GENERALIZATION
DELETE FROM GENERALIZATION
  WHERE CITYOBJECT_ID IN
    (SELECT ID FROM CITYOBJECT
      WHERE CLASS_ID=24
        OR CLASS_ID=25
        OR CLASS_ID=26
        OR CLASS_ID=27
        OR CLASS_ID=28
        OR CLASS_ID=29
        OR CLASS_ID=30
        OR CLASS_ID=31
        OR CLASS_ID=32
        OR CLASS_ID=33
        OR CLASS_ID=34
        OR CLASS_ID=35
        OR CLASS_ID=36
        OR CLASS_ID=37
        OR CLASS_ID=38
        OR CLASS_ID=39
        OR CLASS_ID=40
        OR CLASS_ID=41
        OR CLASS_ID=58);

DELETE FROM GENERALIZATION
  WHERE GENERALIZES_TO_ID IN
    (SELECT ID FROM CITYOBJECT
      WHERE CLASS_ID=24
        OR CLASS_ID=25
        OR CLASS_ID=26
        OR CLASS_ID=27

```



```

        OR CLASS_ID=28
        OR CLASS_ID=29
        OR CLASS_ID=30
        OR CLASS_ID=31
        OR CLASS_ID=32
        OR CLASS_ID=33
        OR CLASS_ID=34
        OR CLASS_ID=35
        OR CLASS_ID=36
        OR CLASS_ID=37
        OR CLASS_ID=38
        OR CLASS_ID=39
        OR CLASS_ID=40
        OR CLASS_ID=41
        OR CLASS_ID=58);

--// *****
--// delete city object groups
--// *****
SELECT 'Deleting city object groups...' as message from DUAL;

--// delete entries from GROUP_TO_CITYOBJECT
DELETE FROM GROUP_TO_CITYOBJECT
    WHERE CITYOBJECT_ID IN
        (SELECT ID FROM CITYOBJECT
         WHERE CLASS_ID=24
          OR CLASS_ID=25
          OR CLASS_ID=26
          OR CLASS_ID=27
          OR CLASS_ID=28
          OR CLASS_ID=29
          OR CLASS_ID=30
          OR CLASS_ID=31
          OR CLASS_ID=32
          OR CLASS_ID=33
          OR CLASS_ID=34
          OR CLASS_ID=35
          OR CLASS_ID=36
          OR CLASS_ID=37
          OR CLASS_ID=38
          OR CLASS_ID=39
          OR CLASS_ID=40
          OR CLASS_ID=41
          OR CLASS_ID=58);

DELETE FROM GROUP_TO_CITYOBJECT
    WHERE CITYOBJECTGROUP_ID IN
        (SELECT ID FROM CITYOBJECTGROUP
         WHERE PARENT_CITYOBJECT_ID IN
            (SELECT ID FROM CITYOBJECT
             WHERE CLASS_ID=24
              OR CLASS_ID=25
              OR CLASS_ID=26
              OR CLASS_ID=27
              OR CLASS_ID=28
              OR CLASS_ID=29
              OR CLASS_ID=30
              OR CLASS_ID=31
              OR CLASS_ID=32
              OR CLASS_ID=33
              OR CLASS_ID=34
              OR CLASS_ID=35
              OR CLASS_ID=36
              OR CLASS_ID=37
              OR CLASS_ID=38
              OR CLASS_ID=39
              OR CLASS_ID=40
              OR CLASS_ID=41
              OR CLASS_ID=58));

ALTER TABLE CITYOBJECTGROUP
    DISABLE
    CONSTRAINT CITYOBJECT_GROUP_FK;

--// delete entries from TEXTUREPARAM
DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
         WHERE ROOT_ID IN
            (SELECT SURFACE_GEOMETRY_ID FROM CITYOBJECTGROUP
             WHERE (SELECT COUNT(*) FROM GROUP_TO_CITYOBJECT
                  WHERE CITYOBJECTGROUP_ID = CITYOBJECTGROUP.ID) =
0));

--// delete entries from SURFACE_GEOMETRY
DELETE FROM SURFACE_GEOMETRY
    WHERE ROOT_ID IN
        (SELECT SURFACE_GEOMETRY_ID FROM CITYOBJECTGROUP
         WHERE (SELECT COUNT(*) FROM GROUP_TO_CITYOBJECT

```

```

WHERE CITYOBJECTGROUP_ID = CITYOBJECTGROUP.ID) = 0);

--// delete entries from CITYOBJECTGROUP
DELETE FROM CITYOBJECTGROUP
WHERE (SELECT COUNT(*) FROM GROUP_TO_CITYOBJECT
WHERE CITYOBJECTGROUP_ID = CITYOBJECTGROUP.ID) = 0;

ALTER TABLE CITYOBJECTGROUP
ENABLE
CONSTRAINT CITYOBJECT_GROUP_FK;

--// *****
--// delete appearances
--// *****
SELECT 'Deleting appearances...' as message from DUAL;

DELETE APPEAR_TO_SURFACE_DATA
WHERE SURFACE_DATA_ID IN
(SELECT ID FROM SURFACE_DATA
WHERE (SELECT COUNT(*) FROM TEXTUREPARAM
WHERE SURFACE_DATA_ID = SURFACE_DATA.ID) = 0);

--// delete entries from SURFACE_DATA
DELETE SURFACE_DATA
WHERE (SELECT COUNT(*) FROM TEXTUREPARAM
WHERE SURFACE_DATA_ID = SURFACE_DATA.ID) = 0;

--// delete entries from APPEARANCE
DELETE APPEARANCE
WHERE (SELECT COUNT(*) FROM APPEAR_TO_SURFACE_DATA
WHERE APPEARANCE_ID = APPEARANCE.ID) = 0;

DELETE FROM APPEARANCE
WHERE CITYOBJECT_ID IN
(SELECT ID FROM CITYOBJECT
WHERE CLASS_ID=24
OR CLASS_ID=25
OR CLASS_ID=26
OR CLASS_ID=27
OR CLASS_ID=28
OR CLASS_ID=29
OR CLASS_ID=30
OR CLASS_ID=31
OR CLASS_ID=32
OR CLASS_ID=33
OR CLASS_ID=34
OR CLASS_ID=35
OR CLASS_ID=36
OR CLASS_ID=37
OR CLASS_ID=38
OR CLASS_ID=39
OR CLASS_ID=40
OR CLASS_ID=41
OR CLASS_ID=58);

--// *****
--// delete city object members
--// *****
SELECT 'Deleting city object members...' as message from DUAL;

--// delete entries from CITYOBJECT_MEMBER
DELETE FROM CITYOBJECT_MEMBER
WHERE CITYOBJECT_ID IN
(SELECT ID FROM CITYOBJECT
WHERE CLASS_ID=24
OR CLASS_ID=25
OR CLASS_ID=26
OR CLASS_ID=27
OR CLASS_ID=28
OR CLASS_ID=29
OR CLASS_ID=30
OR CLASS_ID=31
OR CLASS_ID=32
OR CLASS_ID=33
OR CLASS_ID=34
OR CLASS_ID=35
OR CLASS_ID=36
OR CLASS_ID=37
OR CLASS_ID=38
OR CLASS_ID=39
OR CLASS_ID=40
OR CLASS_ID=41
OR CLASS_ID=58);

--// *****
--// delete city models
--// *****
SELECT 'Deleting city models...' as message from DUAL;

```

```

--// delete entries from APPEARANCE
DELETE FROM APPEARANCE
  WHERE CITYMODEL_ID IN
    (SELECT ID FROM CITYMODEL
     WHERE (SELECT COUNT(*) FROM CITYOBJECT_MEMBER
            WHERE CITYMODEL_ID = CITYMODEL.ID) = 0);

--// delete entries from CITYMODEL
DELETE FROM CITYMODEL
  WHERE (SELECT COUNT(*) FROM CITYOBJECT_MEMBER
         WHERE CITYMODEL_ID = CITYMODEL.ID) = 0;

--// *****
--// delete city objects
--// *****
SELECT 'Deleting city objects...' as message from DUAL;

--// delete entries from CITYOBJECT
DELETE FROM CITYOBJECT
  WHERE CLASS_ID=24
     OR CLASS_ID=25
     OR CLASS_ID=26
     OR CLASS_ID=27
     OR CLASS_ID=28
     OR CLASS_ID=29
     OR CLASS_ID=30
     OR CLASS_ID=31
     OR CLASS_ID=32
     OR CLASS_ID=33
     OR CLASS_ID=34
     OR CLASS_ID=35
     OR CLASS_ID=36
     OR CLASS_ID=37
     OR CLASS_ID=38
     OR CLASS_ID=39
     OR CLASS_ID=40
     OR CLASS_ID=41
     OR CLASS_ID=58;

--// enable FK-constraints on SURFACE_GEOMETRY
ALTER TABLE SURFACE_GEOMETRY
  ENABLE
  CONSTRAINT SURFACE_GEOMETRY_FK
  ENABLE
  CONSTRAINT SURFACE_GEOMETRY_FK1;

SELECT 'Deletion of buildings complete!' as message from DUAL;

```

9.11 Drop database

DROP_DB.sql

```
-- DROP_DB.sql
--
-- Authors:      Prof. Dr. Thomas H. Kolbe <kolbe@igg.tu-berlin.de>
--              Gerhard König <gerhard.koenig@tu-berlin.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--              Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008 Institute for Geodesy and Geoinformation Science,
--              Technische Universität Berlin, Germany
--              http://www.igg.tu-berlin.de
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
-----
-- About:
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description                                     | Author
-- 2.0.2   | 2008-06-28 | disable versioning before dropping              | TKol
-- 2.0.1   | 2008-06-28 | also drop planning manager tables               | TKol
-- 2.0.0   | 2007-11-23 | release version                                 | TKol
--                                              | GKoe
--                                              | CNag
--                                              | ASta
--
-- Disable versioning (if it was enabled before)
@@DISABLEVERSIONING.sql

DROP SEQUENCE ORTHOPHOTO_IMP_SEQ;

DROP SEQUENCE ORTHOPHOTO_RDT_SEQ;

DROP SEQUENCE ORTHOPHOTO_RDT_IMP_SEQ;

DROP SEQUENCE RASTER_REL_IMP_SEQ;

DROP SEQUENCE RASTER_REL_RDT_SEQ;

DROP SEQUENCE RASTER_REL_RDT_IMP_SEQ;

--//DROP FOREIGN KEYS

ALTER TABLE ADDRESS_TO_BUILDING DROP CONSTRAINT "ADDRESS_TO_BUILDING_FK";
ALTER TABLE ADDRESS_TO_BUILDING DROP CONSTRAINT "ADDRESS_TO_BUILDING_ADDRESS_FK";

ALTER TABLE APPEARANCE DROP CONSTRAINT "APPEARANCE_CITYMODEL_FK";
ALTER TABLE APPEARANCE DROP CONSTRAINT "APPEARANCE_CITYOBJECT_FK";

ALTER TABLE APPEAR_TO_SURFACE_DATA DROP CONSTRAINT "APPEAR_TO_SURFACE_DATA_FK1";
ALTER TABLE APPEAR_TO_SURFACE_DATA DROP CONSTRAINT "APPEAR_TO_SURFACE_DATA_FK";

ALTER TABLE BREAKLINE_RELIEF DROP CONSTRAINT "BREAKLINE_RELIEF_FK";

ALTER TABLE BUILDING DROP CONSTRAINT "BUILDING_SURFACE_GEOMETRY_FK";
ALTER TABLE BUILDING DROP CONSTRAINT "BUILDING_SURFACE_GEOMETRY_FK3";
ALTER TABLE BUILDING DROP CONSTRAINT "BUILDING_CITYOBJECT_FK";
ALTER TABLE BUILDING DROP CONSTRAINT "BUILDING_SURFACE_GEOMETRY_FK1";
ALTER TABLE BUILDING DROP CONSTRAINT "BUILDING_SURFACE_GEOMETRY_FK2";
ALTER TABLE BUILDING DROP CONSTRAINT "BUILDING_BUILDING_FK";
ALTER TABLE BUILDING DROP CONSTRAINT "BUILDING_BUILDING_FK1";

ALTER TABLE BUILDING_FURNITURE DROP CONSTRAINT "BUILDING_FURNITURE_FK1";
ALTER TABLE BUILDING_FURNITURE DROP CONSTRAINT "BUILDING_FURNITURE_FK2";
ALTER TABLE BUILDING_FURNITURE DROP CONSTRAINT "BUILDING_FURNITURE_FK";
ALTER TABLE BUILDING_FURNITURE DROP CONSTRAINT "BUILDING_FURNITURE_ROOM_FK";

ALTER TABLE BUILDING_INSTALLATION DROP CONSTRAINT "BUILDING_INSTALLATION_FK3";
ALTER TABLE BUILDING_INSTALLATION DROP CONSTRAINT "BUILDING_INSTALLATION_FK";
ALTER TABLE BUILDING_INSTALLATION DROP CONSTRAINT "BUILDING_INSTALLATION_ROOM_FK";
ALTER TABLE BUILDING_INSTALLATION DROP CONSTRAINT "BUILDING_INSTALLATION_FK4";
ALTER TABLE BUILDING_INSTALLATION DROP CONSTRAINT "BUILDING_INSTALLATION_FK1";
ALTER TABLE BUILDING_INSTALLATION DROP CONSTRAINT "BUILDING_INSTALLATION_FK2";

ALTER TABLE CITYOBJECT DROP CONSTRAINT "CITYOBJECT_OBJECTCLASS_FK";
```

```

ALTER TABLE CITYOBJECTGROUP DROP CONSTRAINT "CITYOBJECT_GROUP_FK";
ALTER TABLE CITYOBJECTGROUP DROP CONSTRAINT "CITYOBJECTGROUP_CITYOBJECT_FK";
ALTER TABLE CITYOBJECTGROUP DROP CONSTRAINT "CITYOBJECTGROUP_CITYOBJECT_FK1";

ALTER TABLE CITYOBJECT_GENERICATTRIB DROP CONSTRAINT "CITYOBJECT_GENERICATTRIB_FK";
ALTER TABLE CITYOBJECT_GENERICATTRIB DROP CONSTRAINT "CITYOBJECT_GENERICATTRIB_FK1";

ALTER TABLE CITYOBJECT_MEMBER DROP CONSTRAINT "CITYOBJECT_MEMBER_CITYMODEL_FK";
ALTER TABLE CITYOBJECT_MEMBER DROP CONSTRAINT "CITYOBJECT_MEMBER_FK";

ALTER TABLE CITY_FURNITURE DROP CONSTRAINT "CITY_FURNITURE_FK";
ALTER TABLE CITY_FURNITURE DROP CONSTRAINT "CITY_FURNITURE_FK1";
ALTER TABLE CITY_FURNITURE DROP CONSTRAINT "CITY_FURNITURE_FK2";
ALTER TABLE CITY_FURNITURE DROP CONSTRAINT "CITY_FURNITURE_FK3";
ALTER TABLE CITY_FURNITURE DROP CONSTRAINT "CITY_FURNITURE_FK4";
ALTER TABLE CITY_FURNITURE DROP CONSTRAINT "CITY_FURNITURE_FK5";
ALTER TABLE CITY_FURNITURE DROP CONSTRAINT "CITY_FURNITURE_FK6";
ALTER TABLE CITY_FURNITURE DROP CONSTRAINT "CITY_FURNITURE_FK7";
ALTER TABLE CITY_FURNITURE DROP CONSTRAINT "CITY_FURNITURE_CITYOBJECT_FK";

ALTER TABLE EXTERNAL_REFERENCE DROP CONSTRAINT "EXTERNAL_REFERENCE_FK";

ALTER TABLE GENERALIZATION DROP CONSTRAINT "GENERALIZATION_FK1";
ALTER TABLE GENERALIZATION DROP CONSTRAINT "GENERALIZATION_FK";

ALTER TABLE GENERIC_CITYOBJECT DROP CONSTRAINT "GENERIC_CITYOBJECT_FK";
ALTER TABLE GENERIC_CITYOBJECT DROP CONSTRAINT "GENERIC_CITYOBJECT_FK1";
ALTER TABLE GENERIC_CITYOBJECT DROP CONSTRAINT "GENERIC_CITYOBJECT_FK2";
ALTER TABLE GENERIC_CITYOBJECT DROP CONSTRAINT "GENERIC_CITYOBJECT_FK3";
ALTER TABLE GENERIC_CITYOBJECT DROP CONSTRAINT "GENERIC_CITYOBJECT_FK4";
ALTER TABLE GENERIC_CITYOBJECT DROP CONSTRAINT "GENERIC_CITYOBJECT_FK5";
ALTER TABLE GENERIC_CITYOBJECT DROP CONSTRAINT "GENERIC_CITYOBJECT_FK6";
ALTER TABLE GENERIC_CITYOBJECT DROP CONSTRAINT "GENERIC_CITYOBJECT_FK7";
ALTER TABLE GENERIC_CITYOBJECT DROP CONSTRAINT "GENERIC_CITYOBJECT_FK8";
ALTER TABLE GENERIC_CITYOBJECT DROP CONSTRAINT "GENERIC_CITYOBJECT_FK9";
ALTER TABLE GENERIC_CITYOBJECT DROP CONSTRAINT "GENERIC_CITYOBJECT_FK10";

ALTER TABLE GROUP_TO_CITYOBJECT DROP CONSTRAINT "GROUP_TO_CITYOBJECT_FK";
ALTER TABLE GROUP_TO_CITYOBJECT DROP CONSTRAINT "GROUP_TO_CITYOBJECT_FK1";

ALTER TABLE IMPLICIT_GEOMETRY DROP CONSTRAINT "IMPLICIT_GEOMETRY_FK";

ALTER TABLE LAND_USE DROP CONSTRAINT "LAND_USE_CITYOBJECT_FK";
ALTER TABLE LAND_USE DROP CONSTRAINT "LAND_USE_SURFACE_GEOMETRY_FK";
ALTER TABLE LAND_USE DROP CONSTRAINT "LAND_USE_SURFACE_GEOMETRY_FK1";
ALTER TABLE LAND_USE DROP CONSTRAINT "LAND_USE_SURFACE_GEOMETRY_FK2";
ALTER TABLE LAND_USE DROP CONSTRAINT "LAND_USE_SURFACE_GEOMETRY_FK3";
ALTER TABLE LAND_USE DROP CONSTRAINT "LAND_USE_SURFACE_GEOMETRY_FK4";

ALTER TABLE MASSPOINT_RELIEF DROP CONSTRAINT "MASSPOINT_RELIEF_FK";

ALTER TABLE OBJECTCLASS DROP CONSTRAINT "OBJECTCLASS_OBJECTCLASS_FK";

ALTER TABLE OPENING DROP CONSTRAINT "OPENING_SURFACE_GEOMETRY_FK1";
ALTER TABLE OPENING DROP CONSTRAINT "OPENING_CITYOBJECT_FK";
ALTER TABLE OPENING DROP CONSTRAINT "OPENING_SURFACE_GEOMETRY_FK";
ALTER TABLE OPENING DROP CONSTRAINT "OPENING_ADDRESS_FK";

ALTER TABLE OPENING_TO_THEM_SURFACE DROP CONSTRAINT "OPENING_TO_THEMATIC_SURFACE_FK";
ALTER TABLE OPENING_TO_THEM_SURFACE DROP CONSTRAINT "OPENING_TO_THEMATIC_SURFAC_FK1";

ALTER TABLE PLANT_COVER DROP CONSTRAINT "PLANT_COVER_FK";
ALTER TABLE PLANT_COVER DROP CONSTRAINT "PLANT_COVER_FK1";
ALTER TABLE PLANT_COVER DROP CONSTRAINT "PLANT_COVER_FK3";
ALTER TABLE PLANT_COVER DROP CONSTRAINT "PLANT_COVER_FK2";
ALTER TABLE PLANT_COVER DROP CONSTRAINT "PLANT_COVER_CITYOBJECT_FK";

ALTER TABLE RELIEF_COMPONENT DROP CONSTRAINT "RELIEF_COMPONENT_CITYOBJECT_FK";

ALTER TABLE RELIEF_FEATURE DROP CONSTRAINT "RELIEF_FEATURE_CITYOBJECT_FK";

ALTER TABLE RELIEF_FEAT_TO_REL_COMP DROP CONSTRAINT "RELIEF_FEAT_TO_REL_COMP_FK";
ALTER TABLE RELIEF_FEAT_TO_REL_COMP DROP CONSTRAINT "RELIEF_FEAT_TO_REL_COMP_FK1";

ALTER TABLE ROOM DROP CONSTRAINT "ROOM_BUILDING_FK";
ALTER TABLE ROOM DROP CONSTRAINT "ROOM_SURFACE_GEOMETRY_FK";
ALTER TABLE ROOM DROP CONSTRAINT "ROOM_CITYOBJECT_FK";

ALTER TABLE SOLITARY_VEGETAT_OBJECT DROP CONSTRAINT "SOLITARY_VEGETAT_OBJECT_FK";
ALTER TABLE SOLITARY_VEGETAT_OBJECT DROP CONSTRAINT "SOLITARY_VEGETAT_OBJECT_FK1";
ALTER TABLE SOLITARY_VEGETAT_OBJECT DROP CONSTRAINT "SOLITARY_VEGETAT_OBJECT_FK2";
ALTER TABLE SOLITARY_VEGETAT_OBJECT DROP CONSTRAINT "SOLITARY_VEGETAT_OBJECT_FK3";
ALTER TABLE SOLITARY_VEGETAT_OBJECT DROP CONSTRAINT "SOLITARY_VEGETAT_OBJECT_FK4";
ALTER TABLE SOLITARY_VEGETAT_OBJECT DROP CONSTRAINT "SOLITARY_VEGETAT_OBJECT_FK5";
ALTER TABLE SOLITARY_VEGETAT_OBJECT DROP CONSTRAINT "SOLITARY_VEGETAT_OBJECT_FK6";
ALTER TABLE SOLITARY_VEGETAT_OBJECT DROP CONSTRAINT "SOLITARY_VEGETAT_OBJECT_FK7";
ALTER TABLE SOLITARY_VEGETAT_OBJECT DROP CONSTRAINT "SOLITARY_VEGETAT_OBJECT_FK8";

```

```

ALTER TABLE SURFACE_GEOMETRY DROP CONSTRAINT "SURFACE_GEOMETRY_FK";
ALTER TABLE SURFACE_GEOMETRY DROP CONSTRAINT "SURFACE_GEOMETRY_FK1";

ALTER TABLE TEXTUREPARAM DROP CONSTRAINT "TEXTUREPARAM_SURFACE_GEOM_FK";
ALTER TABLE TEXTUREPARAM DROP CONSTRAINT "TEXTUREPARAM_SURFACE_DATA_FK";

ALTER TABLE THEMATIC_SURFACE DROP CONSTRAINT "THEMATIC_SURFACE_ROOM_FK";
ALTER TABLE THEMATIC_SURFACE DROP CONSTRAINT "THEMATIC_SURFACE_BUILDING_FK";
ALTER TABLE THEMATIC_SURFACE DROP CONSTRAINT "THEMATIC_SURFACE_FK";
ALTER TABLE THEMATIC_SURFACE DROP CONSTRAINT "THEMATIC_SURFACE_CITYOBJECT_FK";
ALTER TABLE THEMATIC_SURFACE DROP CONSTRAINT "THEMATIC_SURFACE_FK2";
ALTER TABLE THEMATIC_SURFACE DROP CONSTRAINT "THEMATIC_SURFACE_FK1";

ALTER TABLE TIN_RELIEF DROP CONSTRAINT "TIN_RELIEF_SURFACE_GEOMETRY_FK";
ALTER TABLE TIN_RELIEF DROP CONSTRAINT "TIN_RELIEF_RELIEF_COMPONENT_FK";

ALTER TABLE TRAFFIC_AREA DROP CONSTRAINT "TRAFFIC_AREA_CITYOBJECT_FK";
ALTER TABLE TRAFFIC_AREA DROP CONSTRAINT "TRAFFIC_AREA_FK";
ALTER TABLE TRAFFIC_AREA DROP CONSTRAINT "TRAFFIC_AREA_FK1";
ALTER TABLE TRAFFIC_AREA DROP CONSTRAINT "TRAFFIC_AREA_FK2";
ALTER TABLE TRAFFIC_AREA DROP CONSTRAINT "TRAFFIC_AREA_FK3";

ALTER TABLE TRANSPORTATION_COMPLEX DROP CONSTRAINT "TRANSPORTATION_COMPLEX_FK";
ALTER TABLE TRANSPORTATION_COMPLEX DROP CONSTRAINT "TRANSPORTATION_COMPLEX_FK1";
ALTER TABLE TRANSPORTATION_COMPLEX DROP CONSTRAINT "TRANSPORTATION_COMPLEX_FK2";
ALTER TABLE TRANSPORTATION_COMPLEX DROP CONSTRAINT "TRANSPORTATION_COMPLEX_FK3";
ALTER TABLE TRANSPORTATION_COMPLEX DROP CONSTRAINT "TRANSPORTATION_COMPLEX_FK4";

ALTER TABLE WATERBODY DROP CONSTRAINT "WATERBODY_CITYOBJECT_FK";
ALTER TABLE WATERBODY DROP CONSTRAINT "WATERBODY_SURFACE_GEOMETRY_FK1";
ALTER TABLE WATERBODY DROP CONSTRAINT "WATERBODY_SURFACE_GEOMETRY_FK2";
ALTER TABLE WATERBODY DROP CONSTRAINT "WATERBODY_SURFACE_GEOMETRY_FK3";
ALTER TABLE WATERBODY DROP CONSTRAINT "WATERBODY_SURFACE_GEOMETRY_FK4";
ALTER TABLE WATERBODY DROP CONSTRAINT "WATERBODY_SURFACE_GEOMETRY_FK5";
ALTER TABLE WATERBODY DROP CONSTRAINT "WATERBODY_SURFACE_GEOMETRY_FK";

ALTER TABLE WATERBOD_TO_WATERBND_SRF DROP CONSTRAINT "WATERBOD_TO_WATERBND_FK";
ALTER TABLE WATERBOD_TO_WATERBND_SRF DROP CONSTRAINT "WATERBOD_TO_WATERBND_FK1";

ALTER TABLE WATERBOUNDARY_SURFACE DROP CONSTRAINT "WATERBOUNDARY_SRF_CITYOBJ_FK";
ALTER TABLE WATERBOUNDARY_SURFACE DROP CONSTRAINT "WATERBOUNDARY_SURFACE_FK";
ALTER TABLE WATERBOUNDARY_SURFACE DROP CONSTRAINT "WATERBOUNDARY_SURFACE_FK1";
ALTER TABLE WATERBOUNDARY_SURFACE DROP CONSTRAINT "WATERBOUNDARY_SURFACE_FK2";

--//DROP TABLES AND SEQUENCES

DROP TABLE ADDRESS_CASCADE CONSTRAINTS;
DROP SEQUENCE ADDRESS_SEQ;

DROP TABLE ADDRESS_TO_BUILDING_CASCADE CONSTRAINTS;

DROP TABLE APPEARANCE_CASCADE CONSTRAINTS;
DROP SEQUENCE APPEARANCE_SEQ;

DROP TABLE APPEAR_TO_SURFACE_DATA_CASCADE CONSTRAINTS;

DROP TABLE BREAKLINE_RELIEF_CASCADE CONSTRAINTS;

DROP TABLE BUILDING_CASCADE CONSTRAINTS;

DROP TABLE BUILDING_FURNITURE_CASCADE CONSTRAINTS;

DROP TABLE BUILDING_INSTALLATION_CASCADE CONSTRAINTS;

DROP TABLE CITYMODEL_CASCADE CONSTRAINTS;
DROP SEQUENCE CITYMODEL_SEQ;

DROP TABLE CITYOBJECT_CASCADE CONSTRAINTS;

DROP TABLE CITYOBJECTGROUP_CASCADE CONSTRAINTS;

DROP TABLE CITYOBJECT_GENERICATTRIB_CASCADE CONSTRAINTS;
DROP SEQUENCE CITYOBJECT_GENERICATT_SEQ;

DROP TABLE CITYOBJECT_MEMBER_CASCADE CONSTRAINTS;
DROP SEQUENCE CITYOBJECT_SEQ;

DROP TABLE CITY_FURNITURE_CASCADE CONSTRAINTS;

DROP TABLE DATABASE_SRS_CASCADE CONSTRAINTS;

DROP TABLE EXTERNAL_REFERENCE_CASCADE CONSTRAINTS;
DROP SEQUENCE EXTERNAL_REF_SEQ;

DROP TABLE GENERALIZATION_CASCADE CONSTRAINTS;

DROP TABLE GENERIC_CITYOBJECT_CASCADE CONSTRAINTS;

```

```
DROP TABLE GROUP_TO_CITYOBJECT CASCADE CONSTRAINTS;

DROP TABLE IMPLICIT_GEOMETRY CASCADE CONSTRAINTS;
DROP SEQUENCE IMPLICIT_GEOMETRY_SEQ;

DROP TABLE LAND_USE CASCADE CONSTRAINTS;

DROP TABLE MASSPOINT_RELIEF CASCADE CONSTRAINTS;

DROP TABLE OBJECTCLASS CASCADE CONSTRAINTS;

DROP TABLE OPENING CASCADE CONSTRAINTS;

DROP TABLE OPENING_TO_THEM_SURFACE CASCADE CONSTRAINTS;

DROP TABLE PLANT_COVER CASCADE CONSTRAINTS;

DROP TABLE RELIEF CASCADE CONSTRAINTS;

DROP TABLE RELIEF_COMPONENT CASCADE CONSTRAINTS;

DROP TABLE RELIEF_FEATURE CASCADE CONSTRAINTS;

DROP TABLE RELIEF_FEAT_TO_REL_COMP CASCADE CONSTRAINTS;

DROP TABLE ROOM CASCADE CONSTRAINTS;

DROP TABLE SOLITARY_VEGETAT_OBJECT CASCADE CONSTRAINTS;

DROP TABLE SURFACE_DATA CASCADE CONSTRAINTS;

DROP SEQUENCE SURFACE_DATA_SEQ;
DROP TABLE SURFACE_GEOMETRY CASCADE CONSTRAINTS;

DROP SEQUENCE SURFACE_GEOMETRY_SEQ;

DROP TABLE TEXTUREPARAM CASCADE CONSTRAINTS;

DROP TABLE THEMATIC_SURFACE CASCADE CONSTRAINTS;

DROP TABLE TIN_RELIEF CASCADE CONSTRAINTS;

DROP TABLE TRAFFIC_AREA CASCADE CONSTRAINTS;

DROP TABLE TRANSPORTATION_COMPLEX CASCADE CONSTRAINTS;

DROP TABLE WATERBODY CASCADE CONSTRAINTS;

DROP TABLE WATERBOD_TO_WATERBND_SRF CASCADE CONSTRAINTS;

DROP TABLE WATERBOUNDARY_SURFACE CASCADE CONSTRAINTS;

DROP TABLE ORTHOPHOTO_RDT CASCADE CONSTRAINT PURGE;

DROP TABLE ORTHOPHOTO_RDT_IMP CASCADE CONSTRAINT PURGE;

DROP TABLE ORTHOPHOTO_IMP CASCADE CONSTRAINT PURGE;

DROP TABLE ORTHOPHOTO CASCADE CONSTRAINT PURGE;

DROP TABLE RASTER_RELIEF_IMP CASCADE CONSTRAINT PURGE;

DROP TABLE RASTER_RELIEF_RDT CASCADE CONSTRAINT PURGE;

DROP TABLE RASTER_RELIEF_IMP_RDT CASCADE CONSTRAINT PURGE;

DROP TABLE RASTER_RELIEF CASCADE CONSTRAINT PURGE;

DROP TABLE IMPORT_PROCEDURES CASCADE CONSTRAINT PURGE;

@@DROP_PLANNINGMANAGER.sql

@@GEODB_PKG/DROP_GEODB_PKG.sql

PURGE RECYCLEBIN;
```

9.12 Package GEODB

CREATE_GEODB_PKG.sql

```
-- CREATE_GEODB_PKG.sql
--
-- Authors:      Claus Nagel <nagel@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
--
-----
-- About:
-- Creates subpackages "geodb_*".
--
-----
-- ChangeLog:
--
--
-- Version | Date       | Description                               | Author
-- 1.0.0   | 2008-09-10 | release version                           | CNag
--
SELECT 'Creating packages 'geodb_util'', 'geodb_idx'', 'geodb_stat'', and corresponding
types' as message from DUAL;
@@UTIL.sql;
@@IDX.sql;
@@STAT.sql;
SELECT 'Packages 'geodb_util'', 'geodb_idx'', and 'geodb_stat'' created' as message from
DUAL;

SELECT 'Creating matching tool packages 'geodb_match'', 'geodb_process_matches'',
'geodb_delete_by_lineage'', and corresponding types' as message from DUAL;
@@MATCH.sql;
@@PROCESS_MATCHES.sql;
@@DELETE_BY_LINEAGE.sql;
SELECT 'Packages 'geodb_match'', 'geodb_process_matches'', and 'geodb_delete_by_lineage''
created' as message from DUAL;
```


DROP_GEODB_PKG.sql

```
-- DROP_GEODB_PKG.sql
--
-- Authors:      Claus Nagel <nagel@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
-- Drops subpackages "geodb_*".
-----
--
-- ChangeLog:
--
-- Version | Date       | Description          | Author
-- 1.0.0   | 2008-09-10 | release version      | CNag
--
SELECT 'Deleting packages 'geodb_util', 'geodb_idx', 'geodb_stat', and corresponding
types' as message from DUAL;
--// drop global types
DROP TYPE STRARRAY;
DROP TYPE INDEX_OBJ;

--// drop packages
DROP PACKAGE geodb_util;
DROP PACKAGE geodb_idx;
DROP PACKAGE geodb_stat;

SELECT 'Packages 'geodb_util', 'geodb_idx', and 'geodb_stat' deleted' as message from
DUAL;

SELECT 'Deleting matching tool packages 'geodb_match', 'geodb_process_matches',
'geodb_delete_by_lineage', and corresponding types' as message from DUAL;
--// drop packages
DROP PACKAGE geodb_match;
DROP PACKAGE geodb_process_matches;
DROP PACKAGE geodb_delete_by_lineage;

--// drop tables
DROP TABLE match_result;
DROP TABLE match_master_aggr_geom;
DROP TABLE match_cand_aggr_geom;
DROP TABLE match_allocate_geom;
TRUNCATE TABLE match_tmp_building;
DROP TABLE match_tmp_building;

TRUNCATE TABLE MATCH_RESULT_RELEVANT;
DROP TABLE MATCH_RESULT_RELEVANT;
DROP TABLE COLLECT_GEOM;
DROP TABLE CONTAINER_IDS;
SELECT 'Packages 'geodb_match', 'geodb_process_matches', and 'geodb_delete_by_lineage'
deleted' as message from DUAL;
```

GEODB_IDX**IDX.sql**

```
-- IDX.sql
--
-- Authors:      Claus Nagel <nagel@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                           Technische Universität Berlin, Germany
--                           http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
-- Creates package "geodb_idx" containing utility methods for creating/dropping
-- spatial/normal indexes on versioned/unversioned tables.
-----
--
-- ChangeLog:
--
-- Version | Date       | Description                      | Author
-- 1.0.0   | 2008-09-10 | release version                  | CNag
--
/*****
* TYPE INDEX_OBJ
*
* global type to store information relevant to indexes
*****/
CREATE OR REPLACE TYPE INDEX_OBJ AS OBJECT
(
  index_name          VARCHAR2(100),
  table_name          VARCHAR2(100),
  attribute_name      VARCHAR2(100),
  type                NUMBER(1),
  is_3d               NUMBER(1, 0),
  STATIC function construct_spatial_3d
    (index_name VARCHAR2, table_name VARCHAR2, attribute_name VARCHAR2)
  RETURN INDEX_OBJ,
  STATIC function construct_spatial_2d
    (index_name VARCHAR2, table_name VARCHAR2, attribute_name VARCHAR2)
  RETURN INDEX_OBJ,
  STATIC function construct_normal
    (index_name VARCHAR2, table_name VARCHAR2, attribute_name VARCHAR2)
  RETURN INDEX_OBJ
);
/

/*****
* TYPE BODY INDEX_OBJ
*
* constructors for INDEX_OBJ instances
*****/
CREATE OR REPLACE TYPE BODY INDEX_OBJ IS
  STATIC FUNCTION construct_spatial_3d
    (index_name VARCHAR2, table_name VARCHAR2, attribute_name VARCHAR2)
  RETURN INDEX_OBJ IS
  BEGIN
    return INDEX_OBJ(upper(index_name), upper(table_name), upper(attribute_name), 1, 1);
  END;
  STATIC FUNCTION construct_spatial_2d
    (index_name VARCHAR2, table_name VARCHAR2, attribute_name VARCHAR2)
  RETURN INDEX_OBJ IS
  BEGIN
    return INDEX_OBJ(upper(index_name), upper(table_name), upper(attribute_name), 1, 0);
  END;
  STATIC FUNCTION construct_normal
    (index_name VARCHAR2, table_name VARCHAR2, attribute_name VARCHAR2)
  RETURN INDEX_OBJ IS
  BEGIN
    return INDEX_OBJ(upper(index_name), upper(table_name), upper(attribute_name), 0, 0);
  END;
END;
/

/*****
* PACKAGE geodb_idx
*
* utility methods for index handling
*****/
CREATE OR REPLACE PACKAGE geodb_idx
AS
```

```

TYPE index_table IS TABLE OF INDEX_OBJ;
FUNCTION index_status(idx INDEX_OBJ) RETURN VARCHAR2;
FUNCTION create_index(idx INDEX_OBJ, is_versioned BOOLEAN, params VARCHAR2 := '') RETURN
VARCHAR2;
FUNCTION drop_index(idx INDEX_OBJ, is_versioned BOOLEAN) RETURN VARCHAR2;
FUNCTION create_spatial_indexes RETURN STRARRAY;
FUNCTION drop_spatial_indexes RETURN STRARRAY;
FUNCTION create_normal_indexes RETURN STRARRAY;
FUNCTION drop_normal_indexes RETURN STRARRAY;
END geodb_idx;
/

CREATE OR REPLACE PACKAGE BODY geodb_idx
AS
    NORMAL CONSTANT NUMBER(1) := 0;
    SPATIAL CONSTANT NUMBER(1) := 1;

    INDICES CONSTANT index_table := index_table(
        INDEX_OBJ.construct_spatial_3d('CITYOBJECT_SPX', 'CITYOBJECT', 'ENVELOPE'),
        INDEX_OBJ.construct_spatial_3d('SURFACE_GEOM_SPX', 'SURFACE_GEOMETRY', 'GEOMETRY'),
        INDEX_OBJ.construct_normal('CITYOBJECT_INX', 'CITYOBJECT', 'GMLID, GMLID_CODESPACE'),
        INDEX_OBJ.construct_normal('SURFACE_GEOMETRY_INX', 'SURFACE_GEOMETRY', 'GMLID,
GMLID_CODESPACE'),
        INDEX_OBJ.construct_normal('APPEARANCE_INX', 'APPEARANCE', 'GMLID, GMLID_CODESPACE'),
        INDEX_OBJ.construct_normal('SURFACE_DATA_INX', 'SURFACE_DATA', 'GMLID, GMLID_CODESPACE')
    );

    /*****
    * index_status
    *
    * @param idx index to retrieve status from
    * @return VARCHAR2 string representation of status, may include
    *         'DROPPED', 'VALID', 'FAILED', 'INVALID'
    *****/
    FUNCTION index_status(idx INDEX_OBJ) RETURN VARCHAR2
    IS
        status VARCHAR2(20);
    BEGIN
        IF idx.type = SPATIAL THEN
            execute immediate 'SELECT UPPER(DOMIDX_OPSTATUS) FROM USER_INDEXES WHERE INDEX_NAME=:1'
into status using idx.index_name;
        ELSE
            execute immediate 'SELECT UPPER(STATUS) FROM USER_INDEXES WHERE INDEX_NAME=:1' into
status using idx.index_name;
        END IF;

        RETURN status;
    EXCEPTION
        WHEN NO DATA FOUND THEN
            RETURN 'DROPPED';
        WHEN others THEN
            RETURN 'INVALID';
    END;

    /*****
    * create_spatial_metadata
    *
    * @param idx index to create metadata for
    *****/
    PROCEDURE create_spatial_metadata(idx INDEX_OBJ)
    IS
        srid DATABASE_SRS.SRID%TYPE;
    BEGIN
        execute immediate 'DELETE FROM USER_SDO_GEOM_METADATA WHERE TABLE_NAME=:1 AND
COLUMN_NAME=:2' using idx.table_name, idx.attribute_name;
        execute immediate 'select SRID from DATABASE_SRS' into srid;

        IF idx.is_3d = 0 THEN
            execute immediate 'INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO,
SRID)
                                VALUES (:1, :2,
                                    MDSYS.SDO_DIM_ARRAY
                                    (
                                        MDSYS.SDO_DIM_ELEMENT(''X'', 0.000, 10000000.000, 0.0005),
                                        MDSYS.SDO_DIM_ELEMENT(''Y'', 0.000, 10000000.000, 0.0005)), :3
                                    )' using idx.table_name, idx.attribute_name, srid;
        ELSE
            execute immediate 'INSERT INTO USER_SDO_GEOM_METADATA (TABLE_NAME, COLUMN_NAME, DIMINFO,
SRID)
                                VALUES (:1, :2,
                                    MDSYS.SDO_DIM_ARRAY
                                    (
                                        MDSYS.SDO_DIM_ELEMENT(''X'', 0.000, 10000000.000, 0.0005),
                                        MDSYS.SDO_DIM_ELEMENT(''Y'', 0.000, 10000000.000, 0.0005),
                                        MDSYS.SDO_DIM_ELEMENT(''Z'', -1000, 10000, 0.0005)), :3
                                    )' using idx.table_name, idx.attribute_name, srid;
        END IF;
    END;
END;

```

```

/*****
* create_index
*
* @param idx index to create
* @param is_versioned TRUE if database table is version-enabled
* @return VARCHAR2 sql error code, 0 for no errors
*****/
FUNCTION create_index(idx INDEX_OBJ, is_versioned BOOLEAN, params VARCHAR2 := '') RETURN
VARCHAR2
IS
    create_ddl VARCHAR2(1000);
    table_name VARCHAR2(100);
    sql_err_code VARCHAR2(20);
BEGIN
    IF index_status(idx) <> 'VALID' THEN
        sql_err_code := drop_index(idx, is_versioned);

        BEGIN
            table_name := idx.table_name;

            IF is_versioned THEN
                dbms_wm.BEGINDDL(idx.table_name);
                table_name := table_name || '_LTS';
            END IF;

            create_ddl := 'CREATE INDEX ' || idx.index_name || ' ON ' || table_name || '(' ||
idx.attribute_name || ')';

            IF idx.type = SPATIAL THEN
                create_spatial_metadata(idx);
                create_ddl := create_ddl || ' INDEXTYPE IS MDSYS.SPATIAL_INDEX';
            END IF;

            IF params <> '' THEN
                create_ddl := create_ddl || ' ' || params;
            END IF;

            execute immediate create_ddl;

            IF is_versioned THEN
                dbms_wm.COMMITDDL(idx.table_name);
            END IF;

        EXCEPTION
            WHEN others THEN
                dbms_output.put_line(SQLERRM);

                IF is_versioned THEN
                    dbms_wm.ROLLBACKDDL(idx.table_name);
                END IF;

            RETURN SQLCODE;
        END;
    END IF;

    RETURN '0';
END;

/*****
* drop_index
*
* @param idx index to drop
* @param is_versioned TRUE if database table is version-enabled
* @return VARCHAR2 sql error code, 0 for no errors
*****/
FUNCTION drop_index(idx INDEX_OBJ, is_versioned BOOLEAN) RETURN VARCHAR2
IS
    index_name VARCHAR2(100);
BEGIN
    IF index_status(idx) <> 'DROPPED' THEN
        BEGIN
            index_name := idx.index_name;

            IF is_versioned THEN
                dbms_wm.BEGINDDL(idx.table_name);
                index_name := index_name || '_LTS';
            END IF;

            execute immediate 'DROP INDEX ' || index_name;

            IF is_versioned THEN
                dbms_wm.COMMITDDL(idx.table_name);
            END IF;
        EXCEPTION
            WHEN others THEN
                dbms_output.put_line(SQLERRM);

                IF is_versioned THEN
                    dbms_wm.ROLLBACKDDL(idx.table_name);
                END IF;
        END;
    END IF;

    RETURN '0';
END;

```

```

        END IF;

        RETURN SQLCODE;
    END;
END IF;

RETURN '0';
END;

/*****
* create_indexes
* private convience method for invoking create_index on indexes
* of same index type
*
* @param type type of index, e.g. SPATIAL or NORMAL
* @return STRARRAY array of log message strings
*****/
FUNCTION create_indexes(type SMALLINT) RETURN STRARRAY
IS
    log STRARRAY;
    sql_error_code VARCHAR2(20);
BEGIN
    log := STRARRAY();

    FOR i IN INDICES.FIRST .. INDICES.LAST LOOP
        IF INDICES(i).type = type THEN
            sql_error_code := create_index(INDICES(i),
geodb_util.versioning_table(INDICES(i).table_name) = 'ON');
            log.extend;
            log(log.count) := INDICES(i).index_name || ':' || INDICES(i).table_name || ':' ||
INDICES(i).attribute_name || ':' || sql_error_code || ':' || index_status(INDICES(i));
        END IF;
    END LOOP;

    RETURN log;
END;

/*****
* drop_indexes
* private convience method for invoking drop_index on indexes
* of same index type
*
* @param type type of index, e.g. SPATIAL or NORMAL
* @return STRARRAY array of log message strings
*****/
FUNCTION drop_indexes(type SMALLINT) RETURN STRARRAY
IS
    log STRARRAY;
    sql_error_code VARCHAR2(20);
BEGIN
    log := STRARRAY();

    FOR i in INDICES.FIRST .. INDICES.LAST LOOP
        IF INDICES(i).type = type THEN
            sql_error_code := drop_index(INDICES(i),
geodb_util.versioning_table(INDICES(i).table_name) = 'ON');
            log.extend;
            log(log.count) := INDICES(i).index_name || ':' || INDICES(i).table_name || ':' ||
INDICES(i).attribute_name || ':' || sql_error_code || ':' || index_status(INDICES(i));
        END IF;
    END LOOP;

    RETURN log;
END;

/*****
* create_spatial_indexes
* convience method for invoking create_index on all spatial
* indexes
*
* @return STRARRAY array of log message strings
*****/
FUNCTION create_spatial_indexes RETURN STRARRAY
IS
BEGIN
    dbms_output.enable;
    return create_indexes(SPATIAL);
END;

/*****
* drop_spatial_indexes
* convience method for invoking drop_index on all spatial
* indexes
*
* @return STRARRAY array of log message strings
*****/
FUNCTION drop_spatial_indexes RETURN STRARRAY
IS
BEGIN

```

```

    dbms_output.enable;
    return drop_indexes(SPATIAL);
END;

/*****
* create_normal_indexes
* convenience method for invoking create_index on all normal
* indexes
*
* @return STRARRAY array of log message strings
*****/
FUNCTION create_normal_indexes RETURN STRARRAY
IS
BEGIN
    dbms_output.enable;
    return create_indexes(NORMAL);
END;

/*****
* drop_normal_indexes
* convenience method for invoking drop_index on all normal
* indexes
*
* @return STRARRAY array of log message strings
*****/
FUNCTION drop_normal_indexes RETURN STRARRAY
IS
BEGIN
    dbms_output.enable;
    return drop_indexes(NORMAL);
END;
END geodb_idx;
/

```

GEODB_STAT

STAT.sql

```
-- STAT.sql
--
-- Authors:      Prof. Dr. Lutz Pluemer <pluemer@ikg.uni-bonn.de>
--              Dr. Thomas H. Kolbe <kolbe@ikg.uni-bonn.de>
--              Dr. Gerhard Groeger <groeger@ikg.uni-bonn.de>
--              Joerg Schmittwilken <schmittwilken@ikg.uni-bonn.de>
--              Viktor Stroh <stroh@ikg.uni-bonn.de>
--              Dr. Andreas Poth <poth@lat-lon.de>
--              Claus Nagel <nagel@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                  Technische Universität Berlin, Germany
--                  http://www.igg.tu-berlin.de
--              (c) 2004-2006,  Institute for Cartography and Geoinformation,
--                  Universität Bonn, Germany
--                  http://www.ikg.uni-bonn.de
--              (c) 2005-2006,  lat/lon GmbH, Germany
--                  http://www.lat-lon.de--
--
--              This skript is free software under the LGPL Version 2.1.
--              See the GNU Lesser General Public License at
--              http://www.gnu.org/copyleft/lgpl.html
--              for more details.
--
-----
-- About:
-- Creates package "geodb_stat" containing utility methods for creating
-- database statistics.
--
-----
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.1     | 2008-09-10 | release version | CNag
-- 1.0     | 2006-04-03 | release version | LPlu
--                                     TKol
--                                     GGro
--                                     JSch
--                                     VStr
--                                     APot
--
--
/*****
* PACKAGE geodb_stat
*
* utility methods for creating database statistics
*****/
CREATE OR REPLACE PACKAGE geodb_stat
AS
    FUNCTION table_contents RETURN STRARRAY;
END geodb_stat;
/

CREATE OR REPLACE PACKAGE BODY geodb_stat
AS
    /*****
    * versioning_status
    *
    *****/
    FUNCTION table_contents RETURN STRARRAY
    IS
        report STRARRAY := STRARRAY();
        ws VARCHAR2(30);
        cnt NUMBER;
        refreshDate DATE;
        reportDate DATE;
        pa_id PLANNING_ALTERNATIVE.ID%TYPE;
        pa_title PLANNING_ALTERNATIVE.TITLE%TYPE;

    BEGIN
        SELECT SYSDATE INTO reportDate FROM DUAL;
        report.extend; report(report.count) := ('Database Report on 3D City Model - Report date: '
|| TO_CHAR(reportDate, 'DD.MM.YYYY HH24:MI:SS'));
        report.extend; report(report.count) :=
('=====');

        -- Determine current workspace
        ws := DBMS_WM.GetWorkspace;
        report.extend; report(report.count) := ('Current workspace: ' || ws);

        IF ws != 'LIVE' THEN
```

```

-- Get associated planning alternative
SELECT id,title INTO pa_id,pa_title FROM PLANNING_ALTERNATIVE
WHERE workspace_name=ws;
report.extend; report(report.count) := (' (PlanningAlternative ID ' || pa_id || ': ' ||
pa_title || ' " '));

-- Query date of last refresh
SELECT createtime INTO refreshDate
FROM all_workspace_savepoints
WHERE savepoint='refreshed' AND workspace=ws;
report.extend; report(report.count) := ('Last refresh from LIVE workspace: ' ||
TO_CHAR(refreshDate, 'DD.MM.YYYY HH24:MI:SS'));
END IF;
report.extend; report(report.count) := '';

SELECT count(*) INTO cnt FROM citymodel;
report.extend; report(report.count) := ('#CITYMODEL:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM cityobject_member;
report.extend; report(report.count) := ('#CITYOBJECT_MEMBER:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM cityobject;
report.extend; report(report.count) := ('#CITYOBJECT:\t\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM generalization;
report.extend; report(report.count) := ('#GENERALIZATION:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM external_reference;
report.extend; report(report.count) := ('#EXTERNAL_REFERENCE:\t\t\t' || cnt);

-- Geometry
SELECT count(*) INTO cnt FROM implicit_geometry;
report.extend; report(report.count) := ('#IMPLICIT_GEOMETRY:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM surface_geometry;
report.extend; report(report.count) := ('#SURFACE_GEOMETRY:\t\t\t' || cnt);

-- Building
SELECT count(*) INTO cnt FROM address;
report.extend; report(report.count) := ('#ADDRESS:\t\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM address_to_building;
report.extend; report(report.count) := ('#ADDRESS_TO_BUILDING:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM building;
report.extend; report(report.count) := ('#BUILDING:\t\t\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM building_furniture;
report.extend; report(report.count) := ('#BUILDING_FURNITURE:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM building_installation;
report.extend; report(report.count) := ('#BUILDING_INSTALLATION:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM opening;
report.extend; report(report.count) := ('#OPENING:\t\t\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM opening_to_them_surface;
report.extend; report(report.count) := ('#OPENING_TO_THEM_SURFACE:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM room;
report.extend; report(report.count) := ('#ROOM:\t\t\t\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM thematic_surface;
report.extend; report(report.count) := ('#THEMATIC_SURFACE:\t\t\t' || cnt);

-- CityFurniture
SELECT count(*) INTO cnt FROM city_furniture;
report.extend; report(report.count) := ('#CITY_FURNITURE:\t\t\t' || cnt);

-- CityObjectGroup
SELECT count(*) INTO cnt FROM cityobjectgroup;
report.extend; report(report.count) := ('#CITYOBJECTGROUP:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM group_to_cityobject;
report.extend; report(report.count) := ('#GROUP_TO_CITYOBJECT:\t\t\t' || cnt);

-- LandUse
SELECT count(*) INTO cnt FROM land_use;
report.extend; report(report.count) := ('#LAND_USE:\t\t\t\t\t' || cnt);

-- Relief
SELECT count(*) INTO cnt FROM relief_feature;
report.extend; report(report.count) := ('#RELIEF_FEATURE:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM relief_component;
report.extend; report(report.count) := ('#RELIEF_COMPONENT:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM relief_feat_to_rel_comp;
report.extend; report(report.count) := ('#RELIEF_FEAT_TO_REL_COMP:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM tin_relief;
report.extend; report(report.count) := ('#TIN_RELIEF:\t\t\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM breakline_relief;
report.extend; report(report.count) := ('#BREAKLINE_RELIEF:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM masspoint_relief;
report.extend; report(report.count) := ('#MASSPOINT_RELIEF:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM raster_relief;
report.extend; report(report.count) := ('#RASTER_RELIEF:\t\t\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM raster_relief_imp;
report.extend; report(report.count) := ('#RASTER_RELIEF_IMP:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM relief;
report.extend; report(report.count) := ('#RELIEF:\t\t\t\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM orthophoto;
report.extend; report(report.count) := ('#ORTHOPHOTO:\t\t\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM orthophoto_imp;
report.extend; report(report.count) := ('#ORTHOPHOTO_IMP:\t\t\t' || cnt);

```



```

-- Transportation
SELECT count(*) INTO cnt FROM transportation_complex;
report.extend; report(report.count) := ('#TRANSPORTATION_COMPLEX:\t' || cnt);
SELECT count(*) INTO cnt FROM traffic_area;
report.extend; report(report.count) := ('#TRAFFIC_AREA:\t\t\t' || cnt);

-- Vegetation
SELECT count(*) INTO cnt FROM plant_cover;
report.extend; report(report.count) := ('#PLANT_COVER:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM solitary_vegetat_object;
report.extend; report(report.count) := ('#SOLITARY_VEGETAT_OBJECT:\t' || cnt);

-- WaterBody
SELECT count(*) INTO cnt FROM waterbody;
report.extend; report(report.count) := ('#WATERBODY:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM waterboundary_surface;
report.extend; report(report.count) := ('#WATERBOUNDARY_SURFACE:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM waterbod_to_waterbnd_srf;
report.extend; report(report.count) := ('#WATERBOD_TO_WATERBND_SRF:\t' || cnt);

-- GenericCityObject
SELECT count(*) INTO cnt FROM generic_cityobject;
report.extend; report(report.count) := ('#GENERIC_CITYOBJECT:\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM cityobject_genericattrib;
report.extend; report(report.count) := ('#CITYOBJECT_GENERICATTRIB:\t' || cnt);

-- Appearance
SELECT count(*) INTO cnt FROM appearance;
report.extend; report(report.count) := ('#APPEARANCE:\t\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM surface_data;
SELECT count(*) INTO cnt FROM appear_to_surface_data;
report.extend; report(report.count) := ('#APPEAR_TO_SURFACE_DATA:\t' || cnt);
report.extend; report(report.count) := ('#SURFACE_DATA:\t\t\t\t' || cnt);
SELECT count(*) INTO cnt FROM textureparam;
report.extend; report(report.count) := ('#TEXTUREPARAM:\t\t\t\t' || cnt);

RETURN report;
END;

END geodb_stat;
/

```

GEODB_UTIL

UTIL.sql

```
-- UTIL.sql
--
-- Authors:      Claus Nagel <nagel@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                           Technische Universität Berlin, Germany
--                           http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
-- Creates package "geodb_util" containing utility methods for applications
-- and further subpackages. Therefore, "geodb_util" might be a dependency
-- for other packages and/or methods.
-----
--
-- ChangeLog:
--
-- Version | Date       | Description                                | Author
-- 1.0.0   | 2008-09-10 | release version                            | CNag
--
/*****
* TYPE STRARRAY
*
* global type for arrays of strings, e.g. used for log messages
* and reports
*****/
CREATE OR REPLACE TYPE STRARRAY IS TABLE OF VARCHAR2(32767);
/

/*****
* PACKAGE geodb_util
*
* utility methods for applications and subpackages
*****/
CREATE OR REPLACE PACKAGE geodb_util
AS
    FUNCTION versioning_table(table_name VARCHAR2) RETURN VARCHAR2;
    FUNCTION versioning_db RETURN VARCHAR2;
    PROCEDURE db_info(srid OUT DATABASE_SRS.SRID%TYPE, srs OUT DATABASE_SRS.GML_SRS_NAME%TYPE,
versioning OUT VARCHAR2);
    FUNCTION error_msg(err code VARCHAR2) RETURN VARCHAR2;
    FUNCTION split(list VARCHAR2, delim VARCHAR2 := ',') RETURN STRARRAY;
    FUNCTION min(a number, b number) return number;
END geodb_util;
/

CREATE OR REPLACE PACKAGE BODY geodb_util
AS
    /*****
    * versioning_table
    *
    * @param table_name name of the unversioned table, i.e., omit
    *                   suffixes such as _LT
    * @return VARCHAR2 'ON' for version-enabled, 'OFF' otherwise
    *****/
    FUNCTION versioning_table(table_name VARCHAR2) RETURN VARCHAR2
    IS
        status USER_TABLES.STATUS%TYPE;
    BEGIN
        execute immediate 'SELECT STATUS FROM USER_TABLES WHERE TABLE_NAME=:1' into status using
table_name || '_LT';
        RETURN 'ON';
    EXCEPTION
        WHEN others THEN
            RETURN 'OFF';
    END;

    /*****
    * versioning_db
    *
    * @return VARCHAR2 'ON' for version-enabled, 'PARTLY' and 'OFF'
    *****/
    FUNCTION versioning_db RETURN VARCHAR2
    IS
        table_names STRARRAY;

```

```

is_versioned BOOLEAN := FALSE;
not_versioned BOOLEAN := FALSE;
BEGIN
  table_names :=
split('ADDRESS,ADDRESS TO BUILDING,APPEAR TO SURFACE DATA,APPEARANCE,BREAKLINE RELIEF,BUILDING
,BUILDING FURNITURE,BUILDING INSTALLATION,CITY FURNITURE,CITYMODEL,CITYOBJECT,CITYOBJECT GENER
ICATTRIB,CITYOBJECT MEMBER,CITYOBJECTGROUP,EXTERNAL REFERENCE,GENERALIZATION,GENERIC_CITYOBJEC
T,GROUP TO CITYOBJECT,IMPLICIT GEOMETRY,LAND USE,MASSPOINT RELIEF,OPENING,OPENING TO THEM SURF
ACE,PLANT COVER,RELIEF COMPONENT,RELIEF FEAT TO REL_COMP,RELIEF FEATURE,ROOM,SOLITARY VEGETAT
OBJECT,SURFACE_DATA,SURFACE GEOMETRY,TEXTUREPARAM,THEMATIC SURFACE,TIN_RELIEF,TRAFFIC_AREA,TRA
NSPORTATION_COMPLEX,WATERBOD TO WATERBND_SRF,WATERBODY,WATERBOUNDARY_SURFACE');

  FOR i IN table_names.first .. table_names.last LOOP
    IF versioning_table(table_names(i)) = 'ON' THEN
      is_versioned := TRUE;
    ELSE
      not_versioned := TRUE;
    END IF;
  END LOOP;

  IF is_versioned AND NOT not_versioned THEN
    RETURN 'ON';
  ELSIF is_versioned AND not_versioned THEN
    RETURN 'PARTLY';
  ELSE
    RETURN 'OFF';
  END IF;
END;

/*****
* error_msg
*
* @param srid database srid
* @param srs database srs name
* @param versioning database versioning
*****/
PROCEDURE db_info(srid OUT DATABASE_SRS.SRID%TYPE, srs OUT DATABASE_SRS.GML_SRS_NAME%TYPE,
versioning OUT VARCHAR2)
IS
BEGIN
  execute immediate 'SELECT SRID, GML_SRS_NAME from DATABASE_SRS' into srid, srs;
  versioning := versioning_db;
END;

/*****
* error_msg
*
* @param err_code Oracle SQL error code, usually starting with '-',
* e.g. '-06404'
* @return VARCHAR2 corresponding Oracle SQL error message
*****/
FUNCTION error_msg(err_code VARCHAR2) RETURN VARCHAR2
IS
BEGIN
  RETURN SQLERRM(err_code);
END;

/*****
* split
*
* @param list string to be splitted
* @param delim delimiter used for splitting, defaults to ','
* @return STRARRAY array of strings containing split tokens
*****/
FUNCTION split(list VARCHAR2, delim VARCHAR2 := ',') RETURN STRARRAY
IS
  results STRARRAY := STRARRAY();
  idx pls_integer;
  tmp_list VARCHAR2(32767) := list;
BEGIN
  LOOP
    idx := instr(tmp_list,delim);
    IF idx > 0 THEN
      results.extend;
      results(results.count) := substr(tmp_list, 1, idx-1);
      tmp_list := substr(tmp_list, idx + length(delim));
    ELSE
      results.extend;
      results(results.count) := tmp_list;
      EXIT;
    END IF;
  END LOOP;

  RETURN results;
END;

/*****
* min

```

```
*
* @param a first number value
* @param b second number value
* @return NUMBER the smaller of the two input number values
*****/
FUNCTION min(a NUMBER, b NUMBER)
RETURN NUMBER
IS
BEGIN
    IF a < b THEN
        RETURN a;
    ELSE
        RETURN b;
    END IF;
END;

END geodb_util;
/
```

9.13 Matching Tool

GEODB_MATCH

MATCH.sql

```
-- MATCH.sql
--
-- Authors:      Claus Nagel <nagel@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                               Technische Universität Berlin, Germany
--                               http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description | Author
-- 1.0.0   | 2008-09-10 | release version | CNag
--
drop table match_result;
create table match_result
(
  id1 number,
  parent_id1 number,
  root_id1 number,
  areal number,
  lod1 number,
  lineage varchar2(256),
  id2 number,
  parent_id2 number,
  root_id2 number,
  area2 number,
  lod2 number,
  intersection_geometry mdsys.sdo_geometry,
  intersection_area number,
  areal_cov_by_area2 number,
  area2_cov_by_areal number)
nologging;

drop table match_master_aggr_geom;
create table match_master_aggr_geom
(
  id number,
  parent_id number,
  root_id number,
  geometry mdsys.sdo_geometry)
nologging;

drop table match_cand_aggr_geom;
create table match_cand_aggr_geom
(
  id number,
  parent_id number,
  root_id number,
  geometry mdsys.sdo_geometry)
nologging;

drop table match_allocate_geom;
create table match_allocate_geom
(
  id number,
  parent_id number,
  root_id number,
  geometry mdsys.sdo_geometry)
nologging;

truncate table match_tmp_building;
drop table match_tmp_building;
create global temporary table match_tmp_building
(
  id number,
  parent_id number,
  root_id number,
  geometry_id number)
on commit preserve rows;
```

```

set serveroutput on;

CREATE OR REPLACE PACKAGE geodb_match
AS
    procedure create_matching_table(lod_cand number, lineage cityobject.lineage%type, lod_master
number, tolerance number := 0.001, aggregate_building number := 1);

    procedure allocate_cand_building(lod number, lineage cityobject.lineage%type);
    procedure allocate_master_building(lod number, lineage cityobject.lineage%type);
    procedure allocate_geometry(lod number);
    procedure rectify_geometry(tolerance number := 0.001);
    procedure aggregate_geometry(table_name varchar2, tolerance number := 0.001,
aggregate_building number := 1);
    procedure join_cand_master(lod_cand number, lineage cityobject.lineage%type, lod_master
number, tolerance number := 0.001);

    function aggregate_mbr(table_name varchar2) return mdsys.sdo_geometry;
    function aggregate_geometry_by_id(id number, tolerance number := 0.001, aggregate_building
number := 1) return mdsys.sdo_geometry;
    function to_2d(geom mdsys.sdo_geometry) return mdsys.sdo_geometry;
END geodb_match;
/

CREATE OR REPLACE PACKAGE BODY geodb_match
AS
    -- private constants
    CAND_GEOMETRY_TABLE constant string(30) := 'MATCH_CAND_AGGR_GEOM';
    MASTER_GEOMETRY_TABLE constant string(30) := 'MATCH_MASTER_AGGR_GEOM';

    -- declaration of private indexes
    match_master_aggr_geom_spx index obj :=
        index_obj.construct_spatial_3d('match_master_aggr_geom_spx', MASTER_GEOMETRY_TABLE,
'geometry');
    match_cand_aggr_geom_spx index obj :=
        index_obj.construct_spatial_3d('match_cand_aggr_geom_spx', CAND_GEOMETRY_TABLE,
'geometry');
    match_result_spx index obj :=
        index_obj.construct_spatial_3d('match_result_spx', 'match_result',
'intersection_geometry');

    match_allocate_id_idx index obj :=
        index_obj.construct_normal('match_allocate_id_idx', 'match_allocate_geom', 'id');
    match_allocate_root_id_idx index obj :=
        index_obj.construct_normal('match_allocate_root_id_idx', 'match_allocate_geom',
'root_id');

    procedure create_matching_table(lod_cand number, lineage cityobject.lineage%type, lod_master
number, tolerance number := 0.001, aggregate_building number := 1)
    is
        log varchar2(4000);
    begin
        -- gather candidate buildings
        allocate_cand_building(lod_cand, lineage);

        -- gather candidate geometry
        allocate_geometry(lod_cand);

        -- rectify candidate geometry
        rectify_geometry(tolerance);

        -- aggregate candidate geometry
        aggregate_geometry(CAND_GEOMETRY_TABLE, tolerance, aggregate_building);

        -- gather master buildings
        allocate_master_building(lod_master, lineage);

        -- gather master geometry
        allocate_geometry(lod_master);

        -- rectify master geometry
        rectify_geometry(tolerance);

        -- aggregate master geometry
        aggregate_geometry(MASTER_GEOMETRY_TABLE, tolerance, aggregate_building);

        -- fill matching table
        join_cand_master(lod_cand, lineage, lod_master, tolerance);

        commit;
    end;

    procedure allocate_cand_building(lod number, lineage cityobject.lineage%type)
    is
    begin
        -- truncate tmp table
        execute immediate 'truncate table match_tmp_building';

        -- retrieve all building tuples belonging to the specified lineage
        execute immediate 'insert all into match_tmp_building

```

```

        select b.id, b.building_parent_id parent_id, b.building_root_id root_id,
b lod' || to_char(lod) || '_geometry_id geometry_id
        from building b, cityobject c
        where c.id = b.id
              and c.lineage = :1' using lineage;
end;

procedure allocate_master_building(lod number, lineage cityobject.lineage%type)
is
begin
    -- truncate tmp table
    execute immediate 'truncate table match_tmp_building';

    -- retrieve all building tupels not belonging to the specified lineage and
    -- whose mbr is interacting with the aggregated mbr of all candidate building footprint
    execute immediate 'insert all into match_tmp_building
        select b.id, b.building_parent_id parent_id, b.building_root_id root_id,
b lod' || to_char(lod) || '_geometry_id geometry_id
        from building b, cityobject c
        where c.id = b.id
              and c.lineage <> :1
              and sdo_filter(c.envelope, :2) = ''TRUE'''
        using lineage,
        aggregate_mbr(CAND_GEOMETRY_TABLE);
end;

procedure allocate_geometry(lod number)
is
    log varchar2(4000);
begin
    -- first, truncate tmp table
    execute immediate 'truncate table match_allocate_geom';
    log := geodb_idx.drop_index(match_allocate_id_idx, false);
    log := geodb_idx.drop_index(match_allocate_root_id_idx, false);

    -- second, retrieve exterior shell surfaces from building
    execute immediate 'insert all /* append nologging */ into match_allocate_geom
        select bl.id, bl.parent_id, bl.root_id, s.geometry
        from match_tmp_building bl, surface_geometry s
        where s.root_id = bl.geometry_id
              and s.geometry is not null';

    -- for lod > 1 we also have to check surfaces from the tables
    -- building installation and thematic surface
    if lod > 1 then
        -- retrieve surfaces from building installations referencing the identified
        -- building tupels
        execute immediate 'insert all /* append nologging */ into match_allocate_geom
            select bl.id, bl.parent_id, bl.root_id, s.geometry
            from match_tmp_building bl, building_installation i, surface_geometry s
            where i.building_id = bl.id
                  and i.is_external = 1
                  and s.root_id = i.lod' || to_char(lod) || '_geometry_id
                  and s.geometry is not null';

        -- retrieve surfaces from thematic surfaces referencing the identified
        -- building tupels
        execute immediate 'insert all /* append nologging */ into match_allocate_geom
            select bl.id, bl.parent_id, bl.root_id, s.geometry
            from match_tmp_building bl, thematic_surface t, surface_geometry s
            where t.building_id = bl.id
                  and upper(t.type) not in ('INTERIORWALLSURFACE', 'CEILINGSURFACE',
            'FLOOR SURFACE')
                  and s.root_id = t.lod' || to_char(lod) || '_multi_surface_id
                  and s.geometry is not null';
    end if;
    exception
        when others then
            dbms_output.put_line('allocate_geometry: ' || SQLERRM);
    end;

    procedure rectify_geometry(tolerance number := 0.001)
    is
    begin
        -- first, call sdo_util.rectify_geometry
        execute immediate 'update match_allocate_geom set geometry =
sdo_util.rectify_geometry(geometry, :1)
        where sdo_geom.validate_geometry(geometry, :2) <> ''TRUE''' using tolerance,
        tolerance;

        -- second, delete those entries for which sdo_util.rectify_geometry yielded null
        execute immediate 'delete from match_allocate_geom where geometry is null';

        -- third, delete vertical surfaces
        execute immediate 'delete from match_allocate_geom where sdo_geom.sdo_area(geometry, :1)
        <= 0' using tolerance;
    exception
        when others then
            dbms_output.put_line('rectify_geometry: ' || SQLERRM);

```

```

end;

procedure aggregate_geometry(table_name varchar2, tolerance number := 0.001,
aggregate_building number := 1)
is
log varchar2(4000);
begin
-- truncate table
execute immediate 'truncate table '||table_name;

-- drop spatial indexes
if match_cand_aggr_geom_spx.table_name = table_name then
log := geodb_idx.drop_index(match_cand_aggr_geom_spx, false);
else
log := geodb_idx.drop_index(match_master_aggr_geom_spx, false);
end if;

if aggregate_building > 0 then
declare
cursor root_id_cur is
select distinct root_id
from match_tmp_building;

begin
log := geodb_idx.create_index(match_allocate_root_id_idx, false, 'nologging');

for root_id_rec in root_id_cur loop
execute immediate 'insert into '||table_name||' (id, parent_id, root_id, geometry)
values (:1, null, :2, :3)'
using root_id_rec.root_id,
root_id_rec.root_id,
aggregate_geometry_by_id(root_id_rec.root_id, tolerance, 1);
end loop;
end;
else
declare
cursor id_cur is
select distinct id, parent_id, root_id
from match_tmp_building;

begin
log := geodb_idx.create_index(match_allocate_id_idx, false);

for id_rec in id_cur loop
execute immediate 'insert into '||table_name||' (id, parent_id, root_id, geometry)
values (:1, :2, :3, :4)'
using id_rec.id,
id_rec.parent_id,
id_rec.root_id,
aggregate_geometry_by_id(id_rec.id, tolerance, 0);
end loop;
end;
end if;

-- clean up aggregate table
execute immediate 'delete from '||table_name||' where geometry is null';

-- create spatial index
if match_cand_aggr_geom_spx.table_name = table_name then
log := geodb_idx.create_index(match_cand_aggr_geom_spx, false);
else
log := geodb_idx.create_index(match_master_aggr_geom_spx, false);
end if;
end;

procedure join_cand_master(lod_cand number, lineage cityobject.lineage%type, lod_master
number, tolerance number := 0.001)
is
log varchar2(4000);
begin
-- clean environment
execute immediate 'truncate table match_result';
log := geodb_idx.drop_index(match_result_spx, false);

execute immediate 'insert all /*+ append nologging */ into match_result
(id1, parent_id1, root_id1, area1, lod1, lineage,
id2, parent_id2, root_id2, area2, lod2,
intersection_geometry)
select c.id id1, c.parent_id parent_id1, c.root_id root_id1,
sdo_geom.sdo_area(c.geometry, :1) area1, :2, :3,
m.id id2, m.parent_id parent_id2, m.root_id root_id2,
sdo_geom.sdo_area(m.geometry, :4) area2, :5,
sdo_geom.sdo_intersection(c.geometry, m.geometry, :6)
from table(sdo_join(:7, 'geometry', :8, 'geometry',
'mask=INSIDE+CONTAINS+EQUAL+COVERS+COVEREDBY+OVERLAPBDYINTERSECT'))
res, '||CAND_GEOMETRY_TABLE||' c, '||MASTER_GEOMETRY_TABLE||' m
where c.rowid = res.rowid1 and m.rowid = res.rowid2'
using tolerance,
lod_cand,

```



```

        lineage,
        tolerance,
        lod_master,
        tolerance,
        CAND_GEOMETRY_TABLE,
        MASTER_GEOMETRY_TABLE;

        execute immediate 'update match_result set intersection_area =
sdo_geom.sdo_area(intersection_geometry, :1)' using tolerance;
        execute immediate 'delete from match_result where intersection_area = 0';
        execute immediate 'update match_result set area1_cov_by_area2 =
geodb_util.min(intersection_area / area1, 1.0),          area2_cov_by_area1 =
geodb_util.min(intersection_area / area2, 1.0)';

        -- create spatial index on intersection geometry
        log := geodb_idx.create_index(match_result_spx, false);
    end;

    function aggregate_mbr(table_name varchar2)
    return mdsys.sdo_geometry
    is
        aggr_mbr mdsys.sdo_geometry;
    begin
        execute immediate 'select sdo_aggr_mbr(geometry) from '||table_name||' into aggr_mbr;
        return aggr_mbr;
    exception
        when others then
            return null;
    end;

    /*
    * create footprint for building by aggregating (using boolean union) all identified
    * polygons.
    */
    function aggregate_geometry_by_id(id number, tolerance number := 0.001, aggregate_building
number := 1)
    return mdsys.sdo_geometry
    is
        aggr_geom mdsys.sdo_geometry;
        attr string(10);
    begin
        if aggregate_building > 0 then
            attr := 'root_id';
        else
            attr := 'id';
        end if;

        execute immediate 'select sdo_aggr_union(mdsys.sdoaggrtype(aggr_geom, :1))
        from (select sdo_aggr_union(mdsys.sdoaggrtype(aggr_geom, :2)) aggr_geom
        from (select sdo_aggr_union(mdsys.sdoaggrtype(aggr_geom, :3)) aggr_geom
        from (select sdo_aggr_union(mdsys.sdoaggrtype(geometry, :4)) aggr_geom
        from match_allocate_geom
        where '||attr||'=:5
        group by mod(rownum, 1000)
        )
        group by mod (rownum, 100)
        )
        group by mod (rownum, 10)
        )'
        into aggr_geom
        using tolerance,
        tolerance,
        tolerance,
        tolerance,
        id;

        return aggr_geom;
    exception
        when others then
            dbms_output.put_line(id || ': ' || SQLERRM);
            return null;
    end;

    /*
    * code taken from http://forums.oracle.com/forums/thread.jspa?messageID=960492&#960492
    */
    function to_2d (geom mdsys.sdo_geometry)
    return mdsys.sdo_geometry
    is
        geom_2d mdsys.sdo_geometry;
        dim_count integer; -- number of dimensions in layer
        gtype integer; -- geometry type (single digit)
        n_points integer; -- number of points in ordinates array
        n_ordinates integer; -- number of ordinates
        i integer;
        j integer;
        k integer;
        offset integer;

```

```

begin
  -- If the input geometry is null, just return null
  if geom is null then
    return (null);
  end if;

  -- Get the number of dimensions from the gtype
  if length (geom.sdo_gtype) = 4 then
    dim_count := substr (geom.sdo_gtype, 1, 1);
    gtype := substr (geom.sdo_gtype, 4, 1);
  else
    -- Indicate failure
    raise_application_error (-20000, 'Unable to determine dimensionality from gtype');
  end if;

  if dim_count = 2 then
    -- Nothing to do, geometry is already 2D
    return (geom);
  end if;

  -- Construct and prepare the output geometry
  geom_2d := mdsys.sdo_geometry (
    2000+gtype, geom.sdo_srid, geom.sdo_point,
    mdsys.sdo_elem_info_array (), mdsys.sdo_ordinate_array()
  );

  -- Process the point structure
  if geom_2d.sdo_point is not null then
    geom_2d.sdo_point.z := null;
  else
    -- It is not a point
    -- Process the ordinates array

    -- Prepare the size of the output array
    n_points := geom.sdo_ordinates.count / dim_count;
    n_ordinates := n_points * 2;
    geom_2d.sdo_ordinates.extend(n_ordinates);

    -- Copy the ordinates array
    j := geom.sdo_ordinates.first; -- index into input elem_info array
    k := 1; -- index into output ordinate array
    for i in 1..n_points loop
      geom_2d.sdo_ordinates (k) := geom.sdo_ordinates (j); -- copy X
      geom_2d.sdo_ordinates (k+1) := geom.sdo_ordinates (j+1); -- copy Y
      j := j + dim_count;
      k := k + 2;
    end loop;

    -- Process the element info array

    -- Copy the input array into the output array
    geom_2d.sdo_elem_info := geom.sdo_elem_info;

    -- Adjust the offsets
    i := geom_2d.sdo_elem_info.first;
    while i < geom_2d.sdo_elem_info.last loop
      offset := geom_2d.sdo_elem_info(i);
      geom_2d.sdo_elem_info(i) := (offset-1)/dim_count*2+1;
      i := i + 3;
    end loop;
  end if;

  return geom_2d;
end;

END geodb_match;
/

```

GEODB_PROCESS_MATCHES

PROCESS_MATCHES.sql

```
-- PROCESS_MATCHES.sql
--
-- Authors:      Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                           Technische Universität Berlin, Germany
--                           http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description                               | Author
-- 1.0.0   | 2008-09-10 | release version                           | ASta
--
truncate table MATCH_RESULT_RELEVANT;
drop table MATCH_RESULT_RELEVANT;
CREATE GLOBAL TEMPORARY TABLE MATCH_RESULT_RELEVANT
(
  "ID1" NUMBER,
  "PARENT_ID1" NUMBER,
  "ROOT_ID1" NUMBER,
  "LOD1" NUMBER,
  "ID2" NUMBER,
  "PARENT_ID2" NUMBER,
  "ROOT_ID2" NUMBER,
  "LOD2" NUMBER
) ON COMMIT PRESERVE ROWS ;

drop table COLLECT_GEOM;
CREATE TABLE COLLECT_GEOM
(
  "BUILDING_ID" NUMBER,
  "GEOMETRY_ID" NUMBER,
  "CITYOBJECT_ID" NUMBER
);

drop table CONTAINER_IDS;
CREATE TABLE CONTAINER_IDS
(
  "BUILDING_ID" NUMBER,
  "CONTAINER_ID" NUMBER
);

CREATE OR REPLACE PACKAGE GEODB_PROCESS_MATCHES
AS
  -- the important public function
  procedure process_matches(delta1 number, delta2 number, lod_src number, lod_dst number,
name_mode number, delimiter varchar2);

  -- private helper functions
  procedure create_relevant_matches(delta1 number, delta2 number);
  procedure collect_all_geometry(lod number);
  procedure remove_geometry_from_cand(lod number);
  procedure create_and_put_container(lod number, name_mode number, delimiter varchar2);
  procedure move_appearance;
  procedure move_geometry;
  procedure delete_multi_surfaces;
  procedure update_lineage(lineage varchar2);
  procedure cleanup;
END geodb_process_matches;
/

CREATE OR REPLACE PACKAGE BODY GEODB_PROCESS_MATCHES
AS

procedure create_relevant_matches(delta1 number, delta2 number)
is
begin
  -- truncate tmp table
  execute immediate 'truncate table match_result_relevant';
```

```

-- retrieve all match tupels with more than a user-specified percentage of area
coverage
execute immediate 'insert all into match_result_relevant (id1, parent_id1, root_id1,
lod1, id2, parent_id2, root_id2, lod2)
select id1, parent_id1, root_id1, lod1, id2, parent_id2, root_id2, lod2
from match_result
where area1_cov_by_area2 >= :1
and area2_cov_by_area1 >= :2' using delta1, delta2;
exception
when others then
dbms_output.put_line('create_relevant_matches: ' || SQLERRM);
end;

procedure collect_all_geometry(lod number)
is
begin
execute immediate 'truncate table collect_geom';
--execute immediate 'create table collect_geom (building_id number, geometry_id number,
cityobject_id number)';

-- retrieve all building and building part geometry
execute immediate 'insert all into collect_geom
select b.building_root_id, b.lod' || to_char(lod) || '_geometry_id, b.id from
building b, match_result_relevant m
where b.building_root_id = m.id1 and b.lod' || to_char(lod) || '_geometry_id is not
null';

if lod >= 2 then
-- retrieve relevant building installation geometry
execute immediate 'insert all /*+ append nologging */ into collect_geom
select b.building_root_id, i.lod' || to_char(lod) || '_geometry_id, i.id
from match_result_relevant m, building_installation i, building b
where i.building_id = b.id
and b.building_root_id = m.id1
and i.is_external = 1
and i.lod' || to_char(lod) || '_geometry_id is not null';

-- retrieve surfaces from relevant thematic surfaces
execute immediate 'insert all /*+ append nologging */ into collect_geom
select b.building_root_id, t.lod' || to_char(lod) || '_multi_surface_id, t.id
from match_result_relevant m, thematic_surface t, building b
where t.building_id = b.id
and b.building_root_id = m.id1
and t.lod' || to_char(lod) || '_multi_surface_id is not null';
end if;

if lod >= 3 then
-- monster join to retrieve all openings of all thematic surfaces belonging to all
buildings and building parts
execute immediate 'insert all /*+ append nologging */ into collect_geom
select b.building_root_id, o.lod' || to_char(lod) || '_multi_surface_id, o.id
from match_result_relevant m, thematic_surface t, building b, opening o,
opening_to_them_surface ot
where t.building_id = b.id
and b.building_root_id = m.id1
and ot.thematic_surface_id = t.id
and o.id = ot.opening_id
and o.lod' || to_char(lod) || '_multi_surface_id is not null';
end if;

if lod >= 4 then
-- room
execute immediate 'insert all /*+ append nologging */ into collect_geom
select b.building_root_id, r.lod4_geometry_id, r.id
from match_result_relevant m, room r, building b
where r.building_id = b.id
and b.building_root_id = m.id1
and r.lod4_geometry_id is not null';

-- building furniture (in rooms) --bei lod r in f geändert
execute immediate 'insert all /*+ append nologging */ into collect_geom
select b.building_root_id, f.lod4_geometry_id, f.id
from match_result_relevant m, room r, building b, building_furniture f
where r.building_id = b.id
and b.building_root_id = m.id1
and f.room_id = r.id
and f.lod4_geometry_id is not null';

-- retrieve relevant internal (or external) building installation geometry (in rooms)
execute immediate 'insert all /*+ append nologging */ into collect_geom
select b.building_root_id, i.lod4_geometry_id, i.id
from match_result_relevant m, building_installation i, building b, room r
where r.building_id = b.id
and i.room_id = r.id
and b.building_root_id = m.id1
and i.lod4_geometry_id is not null';

```

```

-- retrieve surfaces from relevant thematic surfaces (in rooms)
execute immediate 'insert all /*+ append nologging */ into collect_geom
select b.building_root_id, t.lod4_multi_surface_id, t.id
from match_result_relevant m, thematic_surface t, building b, room r
where r.building_id = b.id
and t.room_id = r.id
and b.building_root_id = m.id1
and t.lod4_multi_surface_id is not null';

-- monster join to retrieve all openings of all thematic surfaces belonging to all
rooms in all buildings and building parts
execute immediate 'insert all /*+ append nologging */ into collect_geom
select b.building_root_id, o.lod4_multi_surface_id, o.id
from match_result_relevant m, thematic_surface t, building b, opening o,
opening_to_them_surface ot, room r
where r.building_id = b.id
and t.room_id = r.id
and b.building_root_id = m.id1
and ot.thematic_surface_id = t.id
and o.id = ot.opening_id
and o.lod4_multi_surface_id is not null';

-- retrieve relevant internal building installation geometry
execute immediate 'insert all /*+ append nologging */ into collect_geom
select b.building_root_id, i.lod4_geometry_id, i.id
from match_result_relevant m, building_installation i, building b
where i.building_id = b.id
and b.building_root_id = m.id1
and i.is_external = 0
and i.lod4_geometry_id is not null';

end if;
exception
when others then
dbms_output.put_line('collect_all_geometry: ' || SQLERRM);
end;

procedure remove_geometry_from_cand(lod number)
is
begin
-- retrieve all building and building part geometry
execute immediate 'update building b
set b.lod'||to_char(lod)||'_geometry_id = null
where b.building_root_id in (select id1 from match_result_relevant)';

if lod >= 2 then
-- retrieve relevant building installation geometry
execute immediate 'update building_installation i
set i.lod'||to_char(lod)||'_geometry_id = null
where i.building_id in (select b.id from building b, match_result_relevant m
where b.building_root_id = m.id1)
and i.is_external = 1';

-- retrieve surfaces from relevant thematic surfaces
execute immediate 'update thematic_surface t
set t.lod'||to_char(lod)||'_multi_surface_id = null
where t.building_id in (select b.id from building b,
match_result_relevant m where b.building_root_id = m.id1)';
end if;

if lod >= 3 then
-- monster join to retrieve all openings of all thematic surfaces belonging to all
buildings and building parts
execute immediate 'update opening o
set o.lod'||to_char(lod)||'_multi_surface_id = null
where o.id in
(select ot.opening_id from match_result_relevant m, thematic_surface t,
building b, opening_to_them_surface ot
where ot.thematic_surface_id = t.id
and t.building_id = b.id
and b.building_root_id = m.id1)';
end if;

if lod >= 4 then
-- room
execute immediate 'update room r
set r.lod4_geometry_id = null
where r.building_id in
(select b.id from match_result_relevant m, building b
where b.building_root_id = m.id1)';

-- building furniture (in rooms) --bei lod r in f geändert
execute immediate 'update building_furniture f
set f.lod4_geometry_id = null
where f.room_id in
(select r.id from match_result_relevant m, room r, building b
where r.building_id = b.id
and b.building_root_id = m.id1)';

-- retrieve relevant internal (or external) building installation geometry (in rooms)

```

```

        execute immediate 'update building_installation i
        set i.lod4_geometry_id = null
        where i.room_id in
            (select r.id from match_result_relevant m, building b, room r
             where r.building_id = b.id
             and b.building_root_id = m.id1)';

        -- retrieve surfaces from relevant thematic surfaces (in rooms)
        execute immediate 'update thematic_surface t
        set t.lod4_multi_surface_id = null
        where t.room_id in
            (select r.id from match_result_relevant m, building b, room r
             where r.building_id = b.id
             and b.building_root_id = m.id1)';

        -- monster join to retrieve all openings of all thematic surfaces belonging to all
        rooms in all buildings and building parts
        execute immediate 'update opening o
        set o.lod4_multi_surface_id = null
        where o.id in
            (select ot.opening_id from match_result_relevant m, thematic_surface t,
            building b, opening_to_them_surface ot, room r
             where r.building_id = b.id
             and t.room_id = r.id
             and b.building_root_id = m.id1
             and ot.thematic_surface_id = t.id)';

        -- retrieve relevant internal building installation geometry
        execute immediate 'update building_installation i
        set i.lod4_geometry_id = null
        where i.is_external = 0
              and i.building_id in
            (select b.id from match_result_relevant m, building b
             where b.building_root_id = m.id1)';

    end if;
exception
    when others then
        dbms_output.put_line('collect_all_geometry: ' || SQLERRM);
end;

procedure create_and_put_container(lod number, name_mode number, delimiter varchar2)
is
begin
    execute immediate 'truncate table container_ids';
    --execute immediate 'create table container_ids (building_id number, container_id
number)';

    declare
        cursor building_id_cur is
            select unique (id1)
            from match_result_relevant;
            --select unique (building_id)
            --from collect_geom;

    begin
        -- go through all affected buildings
        for building_id_rec in building_id_cur loop
            -- create geometry id
            execute immediate 'insert into container_ids (building_id, container_id)
            values (:1, surface_geometry_seq.nextval)'
            using building_id_rec.id1;
            -- create multisurface in table
            execute immediate 'insert into surface_geometry (id, parent_id, root_id,
is_solid, is_composite, is_triangulated, is_xlink, is_reverse, geometry)
            values (surface_geometry_seq.currval, null,
surface_geometry_seq.currval, 0, 0, 0, 0, 0, null)';
            -- set building geometry to new multisurface and process name
            if name_mode=1 then
                -- ignore cand name
                execute immediate 'update building b
                set b.lod'||to_char(lod)||'_geometry_id =
surface_geometry_seq.currval
                where b.id = (select id2 from match_result_relevant where
id1 = :1)'
                using building_id_rec.id1;
            else
                if name_mode=2 then
                    -- replace master name with cand name
                    execute immediate 'update building b
                    set b.lod'||to_char(lod)||'_geometry_id =
surface_geometry_seq.currval,
                    b.name = (select name from building where
id = :1)
                    where b.id = (select id2 from
match_result_relevant where id1 = :2)'

```

```

                                using building_id_rec.id1, building_id_rec.id1;
                    else
                        -- append cand name to master
                        execute immediate 'update building b
                                set b.lod'||to_char(lod)||'_geometry_id =
                                b.name = concat(b.name, concat(:1, (select
                                where b.id = (select id2 from
                                using delimiter, building_id_rec.id1,
                                building_id_rec.id1;
                                end if;
                                end if;
                    end loop;
                end;
            exception
            when others then
                execute immediate 'select :1 as message from dual' using 'create_and_put_container: ' ||
                SQLERRM;
                --dbms_output.put_line('create_and_put_container: ' || SQLERRM);
            end;

procedure move_appearance
is
    texcoord_count number;
begin
    execute immediate 'select count(*)
                        from textureparam tp, surface_geometry sg
                        where tp.texture_coordinates is not null
                        and sg.parent_id in (select geometry_id from
collect_geom)
                        and tp.surface_geometry_id = sg.parent_id' into
    texcoord_count;

    if texcoord_count>0 then
        execute immediate 'SELECT :1 as message from DUAL' using 'Warning: there are
    texcoords applied to multisurfaces that will be lost!';
        end if;

        execute immediate 'insert into textureparam tp (surface_geometry_id,
is_texture_parametrization, world_to_texture, texture_coordinates, surface_data_id)
        values (select sg.id, tp.is_texture_parametrization, tp.world_to_texture,
tp.texture_coordinates, tp.surface_data_id
        from textureparam tp, surface_geometry sg
        where tp.texture_coordinates is null
        and sg.parent_id in (select geometry_id from
collect_geom)
        and tp.surface_geometry_id = sg.parent_id)';

    declare
        cursor theme_cur is
            select a.theme, cg.building_id
            from appearance a, collect_geom cg
            where a.cityobject_id = cg.cityobject_id
            group by a.theme, cg.building_id;

    begin
        -- go through all themes
        for theme_rec in theme_cur loop

            -- 1: neue Appearances in Zielgebäude erzeugen
            execute immediate 'insert into appearance (id, name, name_codespace,
description, theme, citymodel_id, cityobject_id)
            values (appearance_seq.nextval, null, null, null, :1, null, :2);'
            using theme_rec.theme, theme_rec.building_id;

            -- 2: Appear to surface_data auf neue appearances umhängen
            execute immediate 'update appear_to_surface_data asd
            set asd.appearance_id = appearance_seq.currval
            where asd.surface_data_id in
            (select tp.surface_data_id from textureparam tp,
surface_geometry sg, collect_geom cg
            where tp.surface_geometry_id = sg.id
            and sg.root_id = cg.geometry_id
            and cg.building_id = :1)'
            using theme_rec.building_id;

        end loop;
    end;

exception
    when others then
        dbms_output.put_line('create_and_put_container: ' || SQLERRM);
end;

procedure move_geometry

```

```

is
begin
    -- update parent of immediate children of all collected geometries
    execute immediate 'update surface_geometry s
    set s.parent_id = (select c.container_id from container_ids c, collect_geom g
                        where s.parent_id = g.geometry_id and c.building_id =
g.building_id)
                        where s.parent_id in (select geometry_id from collect_geom)';

    -- update all root_ids AND REMOVE SOLID FLAG SINCE WE THROW EVERYTHING INTO A
MULTISURFACE
    execute immediate 'update surface_geometry s
    set s.root_id = (select c.container_id from container_ids c, collect_geom g
                    where s.root_id = g.geometry_id and c.building_id =
g.building_id),
                    s.is_solid = 0
                    where s.root_id in (select geometry_id from collect_geom) and s.root_id<>s.id';
exception
    when others then
        dbms_output.put_line('move_geometry: ' || SQLERRM);
end;

procedure delete_multi_surfaces
is
begin
    execute immediate 'delete from surface_geometry
    where id in (select geometry_id from collect_geom)';
end;

procedure update_lineage(lineage varchar2)
is
begin
    -- execute immediate 'update cityobject
    -- set lineage = :1
    -- where id in (select cityobject_id from collect_geom)'
    -- using lineage;

    execute immediate 'update cityobject c
    set c.lineage = :1
    where c.id in (select b.id from building b, match_result_relevant m where
b.building_root_id = m.id1)'
    using lineage;

    -- if lod >= 2 then
    -- retrieve relevant building installation geometry
    execute immediate 'update cityobject c
    set c.lineage = :1
    where c.id in (select i.id from building_installation i, building b,
match_result_relevant m
                    where i.building_id = b.id
                    and b.building_root_id = m.id1
                    and i.is_external = 1)'
    using lineage;

    -- retrieve surfaces from relevant thematic surfaces
    execute immediate 'update cityobject c
    set c.lineage = :1
    where c.id in (select t.id from thematic_surface t, building b,
match_result_relevant m
                    where t.building_id = b.id
                    and b.building_root_id = m.id1)'
    using lineage;

    -- end if;

    -- if lod >= 3 then
    -- monster join to retrieve all openings of all thematic surfaces belonging to all
buildings and building parts
    execute immediate 'update cityobject c
    set c.lineage = :1
    where c.id in (select o.id from opening o, match_result_relevant m,
thematic_surface t, building b, opening_to_them_surface ot
                    where o.id = ot.opening_id
                    and ot.thematic_surface_id = t.id
                    and t.building_id = b.id
                    and b.building_root_id = m.id1)'
    using lineage;

    -- end if;

    -- if lod >= 4 then
    -- room
    execute immediate 'update cityobject c
    set c.lineage = :1
    where c.id in (select r.id from room r, match_result_relevant m, building b
                    where r.building_id = b.id
                    and b.building_root_id = m.id1)'
    using lineage;

    -- building furniture (in rooms) --bei lod r in f geändert
    execute immediate 'update cityobject c

```



```

        set c.lineage = :1
        where c.id in (select f.id from building_furniture f, match_result_relevant m,
room r, building b
        where f.room_id = r.id
        and r.building_id = b.id
        and b.building_root_id = m.id1)'
        using lineage;

-- retrieve relevant internal (or external) building installation geometry (in rooms)
execute immediate 'update cityobject c
set c.lineage = :1
where c.id in (select i.id from building_installation i, match_result_relevant
m, building b, room r
        where i.room_id = r.id
        and r.building_id = b.id
        and b.building_root_id = m.id1)'
using lineage;

-- retrieve surfaces from relevant thematic surfaces (in rooms)
execute immediate 'update cityobject c
set c.lineage = :1
        where c.id in (select t.id from thematic_surface t,
match_result_relevant m, building b, room r
        where t.room_id = r.id
        and r.building_id = b.id
        and b.building_root_id = m.id1)'
using lineage;

-- monster join to retrieve all openings of all thematic surfaces belonging to all
rooms in all buildings and building parts
execute immediate 'update cityobject c
set c.lineage = :1
        where c.id in (select o.id from opening o, match_result_relevant m,
thematic_surface t, building b, opening_to_them_surface ot, room r
        where o.id = ot.opening_id
        and r.building_id = b.id
        and t.room_id = r.id
        and b.building_root_id = m.id1
        and ot.thematic_surface_id = t.id)'
using lineage;

-- retrieve relevant internal building installation geometry
execute immediate 'update cityobject c
set c.lineage = :1
        where c.id in (select i.id from building_installation i, match_result_relevant m,
building b
        where i.building_id = b.id
        and b.building_root_id = m.id1
        and i.is_external = 0)'
using lineage;

-- end if;
exception
    when others then
        dbms_output.put_line('collect_all_geometry: ' || SQLERRM);
end;

procedure cleanup
is
begin
    -- cleanup
    execute immediate 'truncate table collect_geom';
    execute immediate 'truncate table container_ids';
    --execute immediate 'truncate table match_result_relevant';

exception
    when others then
        dbms_output.put_line('cleanup: ' || SQLERRM);
end;

procedure process_matches(delta1 number, delta2 number, lod_src number, lod_dst number,
name_mode number, delimiter varchar2)
is
begin
    -- our algorithm
    create_relevant_matches(delta1, delta2);
    collect_all_geometry(lod_src);
    remove_geometry_from_cand(lod_src);
    move_appearance();
    create_and_put_container(lod_dst, name_mode, delimiter);
    move_geometry();
    delete_multi_surfaces();

    -- cleanup
    cleanup();

    commit;

```

```
exception
  when others then
    dbms_output.put_line('process_matches: ' || SQLERRM);
end;

END geodb_process_matches;
/
```

GEODB_DELETE_BY_LINEAGE

DELETE_BY_LINEAGE.sql

```
-- PROCESS_MATCHES.sql
--
-- Authors:      Alexandra Stadler <stadler@igg.tu-berlin.de>
--
-- Copyright:    (c) 2007-2008  Institute for Geodesy and Geoinformation Science,
--                           Technische Universität Berlin, Germany
--                           http://www.igg.tu-berlin.de
--
--               This skript is free software under the LGPL Version 2.1.
--               See the GNU Lesser General Public License at
--               http://www.gnu.org/copyleft/lgpl.html
--               for more details.
-----
-- About:
--
--
-----
--
-- ChangeLog:
--
-- Version | Date       | Description                               | Author
-- 1.0.0   | 2008-09-10 | release version                           | ASta

CREATE OR REPLACE PACKAGE GEODB_DELETE_BY_LINEAGE
AS
    procedure delete_buildings(lineage varchar2);
END geodb_delete_by_lineage;
/

CREATE OR REPLACE PACKAGE BODY GEODB_DELETE_BY_LINEAGE
AS
    procedure delete_buildings(lineage varchar2)
    is
    begin
        --// disable FK-constraints on SURFACE_GEOMETRY
        execute immediate 'ALTER TABLE SURFACE_GEOMETRY
            DISABLE
            CONSTRAINT SURFACE_GEOMETRY_FK
            DISABLE
            CONSTRAINT SURFACE_GEOMETRY_FK1';

        --// *****
        --// delete openings
        --// *****
        execute immediate 'SELECT :1 as message from DUAL'
        using 'Deleting openings...';

        execute immediate 'ALTER TABLE OPENING
            DISABLE
            CONSTRAINT OPENING_SURFACE_GEOMETRY_FK
            DISABLE
            CONSTRAINT OPENING_SURFACE_GEOMETRY_FK1';

        --// delete entries from OPENING_TO_THEM_SURFACE
        execute immediate 'DELETE FROM OPENING_TO_THEM_SURFACE where opening_id in (select id from
cityobject where lineage=:1)'
        using lineage;

        --// delete entries from TEXTUREPARAM
        execute immediate 'DELETE FROM TEXTUREPARAM
            WHERE SURFACE_GEOMETRY_ID IN
            (SELECT ID FROM SURFACE_GEOMETRY
            WHERE ROOT_ID IN
            (SELECT LOD3_MULTI_SURFACE_ID FROM OPENING where id in (select
id from cityobject where lineage=:1)))'
        using lineage;

        execute immediate 'DELETE FROM TEXTUREPARAM
            WHERE SURFACE_GEOMETRY_ID IN
            (SELECT ID FROM SURFACE_GEOMETRY
            WHERE ROOT_ID IN
            (SELECT LOD4_MULTI_SURFACE_ID FROM OPENING where id in (select
id from cityobject where lineage=:1)))'
        using lineage;

        --// delete entries from SURFACE_GEOMETRY
```

```

execute immediate 'DELETE FROM SURFACE_GEOMETRY
    WHERE ROOT_ID IN
        (SELECT LOD3_MULTI_SURFACE_ID FROM OPENING where id in (select id from
cityobject where lineage=:1))'
using lineage;

execute immediate 'DELETE FROM SURFACE_GEOMETRY
    WHERE ROOT_ID IN
        (SELECT LOD4_MULTI_SURFACE_ID FROM OPENING where id in (select id from
cityobject where lineage=:1))'
using lineage;

--// delete entries from OPENING
execute immediate 'DELETE FROM OPENING where id in (select id from cityobject where
lineage=:1)'
using lineage;

execute immediate 'ALTER TABLE OPENING
    ENABLE
    CONSTRAINT OPENING_SURFACE_GEOMETRY_FK
    ENABLE
    CONSTRAINT OPENING_SURFACE_GEOMETRY_FK1';

--// *****
--// delete addresses
--// *****
execute immediate 'SELECT :1 as message from DUAL'
using 'Deleting addresses...';

execute immediate 'ALTER TABLE ADDRESS_TO_BUILDING
    DISABLE
    CONSTRAINT ADDRESS_TO_BUILDING_ADDRESS_FK
    DISABLE
    CONSTRAINT ADDRESS_TO_BUILDING_FK';

--// delete entries from ADDRESS
execute immediate 'DELETE FROM ADDRESS where id in
    (select address_id from address_to_building where building_id in
    (select id from cityobject where lineage=:1))'
using lineage;

--// delete entries from ADDRESS_TO_BUILDING;
execute immediate 'DELETE FROM ADDRESS_TO_BUILDING where building_id in
    (select id from cityobject where lineage=:1)'
using lineage;

execute immediate 'ALTER TABLE ADDRESS_TO_BUILDING
    ENABLE
    CONSTRAINT ADDRESS_TO_BUILDING_ADDRESS_FK
    ENABLE
    CONSTRAINT ADDRESS_TO_BUILDING_FK';

--// *****
--// delete thematic surfaces
--// *****
execute immediate 'SELECT :1 as message from DUAL'
using 'Deleting thematic surfaces...';

execute immediate 'ALTER TABLE THEMATIC_SURFACE
    DISABLE
    CONSTRAINT THEMATIC_SURFACE_FK
    DISABLE
    CONSTRAINT THEMATIC_SURFACE_FK1
    DISABLE
    CONSTRAINT THEMATIC_SURFACE_FK2';

--// delete entries from TEXTUREPARAM
execute immediate 'DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
            WHERE ROOT_ID IN
                (SELECT LOD2_MULTI_SURFACE_ID FROM THEMATIC_SURFACE where id in
(select id from cityobject where lineage=:1)))'
using lineage;

execute immediate 'DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
            WHERE ROOT_ID IN
                (SELECT LOD3_MULTI_SURFACE_ID FROM THEMATIC_SURFACE where id in
(select id from cityobject where lineage=:1)))'
using lineage;

execute immediate 'DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
            WHERE ROOT_ID IN
                (SELECT LOD4_MULTI_SURFACE_ID FROM THEMATIC_SURFACE where id in
(select id from cityobject where lineage=:1)))'

```

```

using lineage;

--// delete entries from SURFACE_GEOMETRY
execute immediate 'DELETE FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN
(SELECT LOD2_MULTI_SURFACE_ID FROM THEMATIC_SURFACE where id in (select id from
cityobject where lineage=:1))'
using lineage;

execute immediate 'DELETE FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN
(SELECT LOD3_MULTI_SURFACE_ID FROM THEMATIC_SURFACE where id in (select id from
cityobject where lineage=:1))'
using lineage;

execute immediate 'DELETE FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN
(SELECT LOD4_MULTI_SURFACE_ID FROM THEMATIC_SURFACE where id in (select id from
cityobject where lineage=:1))'
using lineage;

--// delete entries from THEMATIC_SURFACE
execute immediate 'DELETE FROM THEMATIC_SURFACE where id in (select id from cityobject where
lineage=:1)'
using lineage;

execute immediate 'ALTER TABLE THEMATIC_SURFACE
ENABLE
CONSTRAINT THEMATIC_SURFACE_FK
ENABLE
CONSTRAINT THEMATIC_SURFACE_FK1
ENABLE
CONSTRAINT THEMATIC_SURFACE_FK2';

--// *****
--// delete building installations
--// *****
execute immediate 'SELECT :1 as message from DUAL'
using 'Deleting building installations...';

execute immediate 'ALTER TABLE BUILDING_INSTALLATION
DISABLE
CONSTRAINT BUILDING_INSTALLATION_FK2
DISABLE
CONSTRAINT BUILDING_INSTALLATION_FK3
DISABLE
CONSTRAINT BUILDING_INSTALLATION_FK4';

--// delete entries from TEXTUREPARAM
execute immediate 'DELETE FROM TEXTUREPARAM
WHERE SURFACE_GEOMETRY_ID IN
(SELECT ID FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN
(SELECT LOD2_GEOMETRY_ID FROM BUILDING_INSTALLATION where id in
(select id from cityobject where lineage=:1)))'
using lineage;

execute immediate 'DELETE FROM TEXTUREPARAM
WHERE SURFACE_GEOMETRY_ID IN
(SELECT ID FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN
(SELECT LOD3_GEOMETRY_ID FROM BUILDING_INSTALLATION where id in
(select id from cityobject where lineage=:1)))'
using lineage;

execute immediate 'DELETE FROM TEXTUREPARAM
WHERE SURFACE_GEOMETRY_ID IN
(SELECT ID FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN
(SELECT LOD4_GEOMETRY_ID FROM BUILDING_INSTALLATION where id in
(select id from cityobject where lineage=:1)))'
using lineage;

--// delete entries from SURFACE_GEOMETRY
execute immediate 'DELETE FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN
(SELECT LOD2_GEOMETRY_ID FROM BUILDING_INSTALLATION where id in (select id from
cityobject where lineage=:1))'
using lineage;

execute immediate 'DELETE FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN
(SELECT LOD3_GEOMETRY_ID FROM BUILDING_INSTALLATION where id in (select id from
cityobject where lineage=:1))'
using lineage;

execute immediate 'DELETE FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN

```

```

        (SELECT LOD4_GEOMETRY_ID FROM BUILDING_INSTALLATION where id in (select id from
cityobject where lineage=:1))'
using lineage;

--// delete entries from BUILDING_INSTALLATION
execute immediate 'DELETE FROM BUILDING_INSTALLATION where id in (select id from cityobject
where lineage=:1)'
using lineage;

execute immediate 'ALTER TABLE BUILDING_INSTALLATION
    ENABLE
    CONSTRAINT BUILDING_INSTALLATION_FK2
    ENABLE
    CONSTRAINT BUILDING_INSTALLATION_FK3
    ENABLE
    CONSTRAINT BUILDING_INSTALLATION_FK4';

--// *****
--// delete building furniture
--// *****
execute immediate 'SELECT :1 as message from DUAL'
using 'Deleting building furniture...';

execute immediate 'ALTER TABLE IMPLICIT_GEOMETRY
    DISABLE
    CONSTRAINT IMPLICIT_GEOMETRY_FK';

--// delete entries from TEXTUREPARAM
execute immediate 'DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
            WHERE ROOT_ID IN
                (SELECT RELATIVE_GEOMETRY_ID FROM IMPLICIT_GEOMETRY
                    WHERE ID IN
                        (SELECT LOD4_IMPLICIT_REP_ID FROM
BUILDING_FURNITURE where id in (select id from cityobject where lineage=:1))))'
using lineage;

--// delete entries from SURFACE_GEOMETRY
execute immediate 'DELETE FROM SURFACE_GEOMETRY
    WHERE ROOT_ID IN
        (SELECT RELATIVE_GEOMETRY_ID FROM IMPLICIT_GEOMETRY
            WHERE ID IN
                (SELECT LOD4_IMPLICIT_REP_ID FROM BUILDING_FURNITURE where id in
(select id from cityobject where lineage=:1))))'
using lineage;

execute immediate 'ALTER TABLE BUILDING_FURNITURE
    DISABLE
    CONSTRAINT BUILDING_FURNITURE_FK
    DISABLE
    CONSTRAINT BUILDING_FURNITURE_FK2';

--// delete entries from IMPLICIT_GEOMETRY
execute immediate 'DELETE FROM IMPLICIT_GEOMETRY
    WHERE ID IN
        (SELECT LOD4_IMPLICIT_REP_ID FROM BUILDING_FURNITURE where id in (select id
from cityobject where lineage=:1))'
using lineage;

--// delete entries from TEXTUREPARAM
execute immediate 'DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
            WHERE ROOT_ID IN
                (SELECT LOD4_GEOMETRY_ID FROM BUILDING_FURNITURE where id in
(select id from cityobject where lineage=:1))))'
using lineage;

--// delete entries from SURFACE_GEOMETRY
execute immediate 'DELETE FROM SURFACE_GEOMETRY
    WHERE ROOT_ID IN
        (SELECT LOD4_GEOMETRY_ID FROM BUILDING_FURNITURE where id in (select id from
cityobject where lineage=:1))'
using lineage;

--// delete entries from BUILDING_FURNITURE
execute immediate 'DELETE FROM BUILDING_FURNITURE where id in (select id from cityobject where
lineage=:1)'
using lineage;

execute immediate 'ALTER TABLE BUILDING_FURNITURE
    ENABLE
    CONSTRAINT BUILDING_FURNITURE_FK
    ENABLE
    CONSTRAINT BUILDING_FURNITURE_FK2';

execute immediate 'ALTER TABLE IMPLICIT_GEOMETRY
    ENABLE

```

```

        CONSTRAINT IMPLICIT_GEOMETRY_FK';

--// *****
--// delete rooms
--// *****
execute immediate 'SELECT :1 as message from DUAL'
using 'Deleting rooms...';

execute immediate 'ALTER TABLE ROOM
    DISABLE
    CONSTRAINT ROOM_SURFACE_GEOMETRY_FK';

--// delete entries from TEXTUREPARAM
execute immediate 'DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
            WHERE ROOT_ID IN
                (SELECT LOD4_GEOMETRY_ID FROM ROOM where id in (select id from
cityobject where lineage=:1)))'
using lineage;

--// delete entries from SURFACE_GEOMETRY
execute immediate 'DELETE FROM SURFACE_GEOMETRY
    WHERE ROOT_ID IN
        (SELECT LOD4_GEOMETRY_ID FROM ROOM where id in (select id from cityobject where
lineage=:1))'
using lineage;

--// delete entries from ROOM
execute immediate 'DELETE FROM ROOM where id in (select id from cityobject where lineage=:1)'
using lineage;

execute immediate 'ALTER TABLE ROOM
    ENABLE
    CONSTRAINT ROOM_SURFACE_GEOMETRY_FK';

--// *****
--// delete buildings
--// *****
execute immediate 'SELECT :1 as message from DUAL'
using 'Deleting buildings...';

execute immediate 'ALTER TABLE BUILDING
    DISABLE
    CONSTRAINT BUILDING_BUILDING_FK
    DISABLE
    CONSTRAINT BUILDING_BUILDING_FK1
    DISABLE
    CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK
    DISABLE
    CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK1
    DISABLE
    CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK2
    DISABLE
    CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK3';

--// delete entries from TEXTUREPARAM
execute immediate 'DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
            WHERE ROOT_ID IN
                (SELECT LOD1_GEOMETRY_ID FROM BUILDING where id in (select id
from cityobject where lineage=:1)))'
using lineage;

execute immediate 'DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
            WHERE ROOT_ID IN
                (SELECT LOD2_GEOMETRY_ID FROM BUILDING where id in (select id
from cityobject where lineage=:1)))'
using lineage;

execute immediate 'DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
            WHERE ROOT_ID IN
                (SELECT LOD3_GEOMETRY_ID FROM BUILDING where id in (select id
from cityobject where lineage=:1)))'
using lineage;

execute immediate 'DELETE FROM TEXTUREPARAM
    WHERE SURFACE_GEOMETRY_ID IN
        (SELECT ID FROM SURFACE_GEOMETRY
            WHERE ROOT_ID IN
                (SELECT LOD4_GEOMETRY_ID FROM BUILDING where id in (select id
from cityobject where lineage=:1)))'
using lineage;

```

```

--// delete entries from SURFACE_GEOMETRY
execute immediate 'DELETE FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN
(SELECT LOD1_GEOMETRY_ID FROM BUILDING where id in (select id from cityobject
where lineage=:1))'
using lineage;

execute immediate 'DELETE FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN
(SELECT LOD2_GEOMETRY_ID FROM BUILDING where id in (select id from cityobject
where lineage=:1))'
using lineage;

execute immediate 'DELETE FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN
(SELECT LOD3_GEOMETRY_ID FROM BUILDING where id in (select id from cityobject
where lineage=:1))'
using lineage;

execute immediate 'DELETE FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN
(SELECT LOD4_GEOMETRY_ID FROM BUILDING where id in (select id from cityobject
where lineage=:1))'
using lineage;

--// delete entries from BUILDING
execute immediate 'DELETE FROM BUILDING where id in (select id from cityobject where
lineage=:1)'
using lineage;

execute immediate 'ALTER TABLE BUILDING
ENABLE
CONSTRAINT BUILDING_BUILDING_FK
ENABLE
CONSTRAINT BUILDING_BUILDING_FK1
ENABLE
CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK
ENABLE
CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK1
ENABLE
CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK2
ENABLE
CONSTRAINT BUILDING_SURFACE_GEOMETRY_FK3';

--// *****
--// delete external references
--// *****
execute immediate 'SELECT :1 as message from DUAL'
using 'Deleting external references...';

--// delete entries from EXTERNAL_REFERENCE
execute immediate 'DELETE FROM EXTERNAL_REFERENCE
WHERE CITYOBJECT_ID IN
(SELECT ID FROM CITYOBJECT
WHERE (CLASS_ID=24
OR CLASS_ID=25
OR CLASS_ID=26
OR CLASS_ID=27
OR CLASS_ID=28
OR CLASS_ID=29
OR CLASS_ID=30
OR CLASS_ID=31
OR CLASS_ID=32
OR CLASS_ID=33
OR CLASS_ID=34
OR CLASS_ID=35
OR CLASS_ID=36
OR CLASS_ID=37
OR CLASS_ID=38
OR CLASS_ID=39
OR CLASS_ID=40
OR CLASS_ID=41
OR CLASS_ID=58)
and lineage=:1)'
using lineage;

--// *****
--// delete generic attributes
--// *****
execute immediate 'SELECT :1 as message from DUAL'
using 'Deleting generic attributes...';

execute immediate 'ALTER TABLE CITYOBJECT_GENERICATTRIB
DISABLE
CONSTRAINT CITYOBJECT_GENERICATTRIB_FK1';

--// delete entries from TEXTUREPARAM
execute immediate 'DELETE FROM TEXTUREPARAM
WHERE SURFACE_GEOMETRY_ID IN

```



```

(SELECT ID FROM SURFACE_GEOMETRY
 WHERE ROOT_ID IN
   (SELECT SURFACE_GEOMETRY_ID FROM CITYOBJECT_GENERICATTRIB
    WHERE CITYOBJECT_ID IN
      (SELECT ID FROM CITYOBJECT
       WHERE (CLASS_ID=24
            OR CLASS_ID=25
            OR CLASS_ID=26
            OR CLASS_ID=27
            OR CLASS_ID=28
            OR CLASS_ID=29
            OR CLASS_ID=30
            OR CLASS_ID=31
            OR CLASS_ID=32
            OR CLASS_ID=33
            OR CLASS_ID=34
            OR CLASS_ID=35
            OR CLASS_ID=36
            OR CLASS_ID=37
            OR CLASS_ID=38
            OR CLASS_ID=39
            OR CLASS_ID=40
            OR CLASS_ID=41
            OR CLASS_ID=58)
       and lineage=:1))) '

using lineage;

--// delete entries from SURFACE_GEOMETRY
execute immediate 'DELETE FROM SURFACE_GEOMETRY
 WHERE ROOT_ID IN
   (SELECT SURFACE_GEOMETRY_ID FROM CITYOBJECT_GENERICATTRIB
    WHERE CITYOBJECT_ID IN
      (SELECT ID FROM CITYOBJECT
       WHERE (CLASS_ID=24
            OR CLASS_ID=25
            OR CLASS_ID=26
            OR CLASS_ID=27
            OR CLASS_ID=28
            OR CLASS_ID=29
            OR CLASS_ID=30
            OR CLASS_ID=31
            OR CLASS_ID=32
            OR CLASS_ID=33
            OR CLASS_ID=34
            OR CLASS_ID=35
            OR CLASS_ID=36
            OR CLASS_ID=37
            OR CLASS_ID=38
            OR CLASS_ID=39
            OR CLASS_ID=40
            OR CLASS_ID=41
            OR CLASS_ID=58)
       and lineage=:1)) '

using lineage;

--// delete entries from CITYOBJECT_GENERICATTRIB
execute immediate 'DELETE FROM CITYOBJECT_GENERICATTRIB
 WHERE CITYOBJECT_ID IN
   (SELECT ID FROM CITYOBJECT
    WHERE (CLASS_ID=24
         OR CLASS_ID=25
         OR CLASS_ID=26
         OR CLASS_ID=27
         OR CLASS_ID=28
         OR CLASS_ID=29
         OR CLASS_ID=30
         OR CLASS_ID=31
         OR CLASS_ID=32
         OR CLASS_ID=33
         OR CLASS_ID=34
         OR CLASS_ID=35
         OR CLASS_ID=36
         OR CLASS_ID=37
         OR CLASS_ID=38
         OR CLASS_ID=39
         OR CLASS_ID=40
         OR CLASS_ID=41
         OR CLASS_ID=58)
    and lineage=:1) '

using lineage;

execute immediate 'ALTER TABLE CITYOBJECT_GENERICATTRIB
 ENABLE
 CONSTRAINT CITYOBJECT_GENERICATTRIB_FK1';

--// *****
--// delete generalizations
--// *****
execute immediate 'SELECT :1 as message from DUAL'

```

```

using 'Deleting generalization hierarchies...';

--// delete entries from GENERALIZATION
execute immediate 'DELETE FROM GENERALIZATION
WHERE CITYOBJECT_ID IN
(SELECT ID FROM CITYOBJECT
WHERE (CLASS_ID=24
OR CLASS_ID=25
OR CLASS_ID=26
OR CLASS_ID=27
OR CLASS_ID=28
OR CLASS_ID=29
OR CLASS_ID=30
OR CLASS_ID=31
OR CLASS_ID=32
OR CLASS_ID=33
OR CLASS_ID=34
OR CLASS_ID=35
OR CLASS_ID=36
OR CLASS_ID=37
OR CLASS_ID=38
OR CLASS_ID=39
OR CLASS_ID=40
OR CLASS_ID=41
OR CLASS_ID=58)
and lineage=:1)'

using lineage;

execute immediate 'DELETE FROM GENERALIZATION
WHERE GENERALIZES_TO_ID IN
(SELECT ID FROM CITYOBJECT
WHERE (CLASS_ID=24
OR CLASS_ID=25
OR CLASS_ID=26
OR CLASS_ID=27
OR CLASS_ID=28
OR CLASS_ID=29
OR CLASS_ID=30
OR CLASS_ID=31
OR CLASS_ID=32
OR CLASS_ID=33
OR CLASS_ID=34
OR CLASS_ID=35
OR CLASS_ID=36
OR CLASS_ID=37
OR CLASS_ID=38
OR CLASS_ID=39
OR CLASS_ID=40
OR CLASS_ID=41
OR CLASS_ID=58)
and lineage=:1)'

using lineage;

--// *****
--// delete city object groups
--// *****
execute immediate 'SELECT :1 as message from DUAL'
using 'Deleting city object groups...';

--// delete entries from GROUP_TO_CITYOBJECT
execute immediate 'DELETE FROM GROUP_TO_CITYOBJECT
WHERE CITYOBJECT_ID IN
(SELECT ID FROM CITYOBJECT
WHERE (CLASS_ID=24
OR CLASS_ID=25
OR CLASS_ID=26
OR CLASS_ID=27
OR CLASS_ID=28
OR CLASS_ID=29
OR CLASS_ID=30
OR CLASS_ID=31
OR CLASS_ID=32
OR CLASS_ID=33
OR CLASS_ID=34
OR CLASS_ID=35
OR CLASS_ID=36
OR CLASS_ID=37
OR CLASS_ID=38
OR CLASS_ID=39
OR CLASS_ID=40
OR CLASS_ID=41
OR CLASS_ID=58)
and lineage=:1)'

using lineage;

execute immediate 'DELETE FROM GROUP_TO_CITYOBJECT
WHERE CITYOBJECTGROUP_ID IN
(SELECT ID FROM CITYOBJECTGROUP
WHERE PARENT_CITYOBJECT_ID IN

```

```

        (SELECT ID FROM CITYOBJECT
         WHERE (CLASS_ID=24
              OR CLASS_ID=25
              OR CLASS_ID=26
              OR CLASS_ID=27
              OR CLASS_ID=28
              OR CLASS_ID=29
              OR CLASS_ID=30
              OR CLASS_ID=31
              OR CLASS_ID=32
              OR CLASS_ID=33
              OR CLASS_ID=34
              OR CLASS_ID=35
              OR CLASS_ID=36
              OR CLASS_ID=37
              OR CLASS_ID=38
              OR CLASS_ID=39
              OR CLASS_ID=40
              OR CLASS_ID=41
              OR CLASS_ID=58)
         and lineage=:1))'

using lineage;

execute immediate 'ALTER TABLE CITYOBJECTGROUP
DISABLE
CONSTRAINT CITYOBJECT_GROUP_FK';

--// delete entries from TEXTUREPARAM
execute immediate 'DELETE FROM TEXTUREPARAM
WHERE SURFACE_GEOMETRY_ID IN
(SELECT ID FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN
(SELECT SURFACE_GEOMETRY_ID FROM CITYOBJECTGROUP
WHERE (SELECT COUNT(*) FROM GROUP_TO_CITYOBJECT
WHERE CITYOBJECTGROUP_ID = CITYOBJECTGROUP.ID) =
0))';

--// delete entries from SURFACE_GEOMETRY
execute immediate 'DELETE FROM SURFACE_GEOMETRY
WHERE ROOT_ID IN
(SELECT SURFACE_GEOMETRY_ID FROM CITYOBJECTGROUP
WHERE (SELECT COUNT(*) FROM GROUP_TO_CITYOBJECT
WHERE CITYOBJECTGROUP_ID = CITYOBJECTGROUP.ID) = 0)';

--// delete entries from CITYOBJECTGROUP
execute immediate 'DELETE FROM CITYOBJECTGROUP
WHERE (SELECT COUNT(*) FROM GROUP_TO_CITYOBJECT
WHERE CITYOBJECTGROUP_ID = CITYOBJECTGROUP.ID) = 0';

execute immediate 'ALTER TABLE CITYOBJECTGROUP
ENABLE
CONSTRAINT CITYOBJECT_GROUP_FK';

--// *****
--// delete appearances
--// *****
execute immediate 'SELECT :1 as message from DUAL'
using 'Deleting appearances...';

execute immediate 'DELETE APPEAR_TO_SURFACE_DATA
WHERE SURFACE_DATA_ID IN
(SELECT ID FROM SURFACE_DATA
WHERE (SELECT COUNT(*) FROM TEXTUREPARAM
WHERE SURFACE_DATA_ID = SURFACE_DATA.ID) = 0)';

--// delete entries from SURFACE_DATA
execute immediate 'DELETE SURFACE_DATA
WHERE (SELECT COUNT(*) FROM TEXTUREPARAM
WHERE SURFACE_DATA_ID = SURFACE_DATA.ID) = 0';

--// delete entries from APPEARANCE
execute immediate 'DELETE APPEARANCE
WHERE (SELECT COUNT(*) FROM APPEAR_TO_SURFACE_DATA
WHERE APPEARANCE_ID = APPEARANCE.ID) = 0';

execute immediate 'DELETE FROM APPEARANCE
WHERE CITYOBJECT_ID IN
(SELECT ID FROM CITYOBJECT
WHERE (CLASS_ID=24
      OR CLASS_ID=25
      OR CLASS_ID=26
      OR CLASS_ID=27
      OR CLASS_ID=28
      OR CLASS_ID=29
      OR CLASS_ID=30
      OR CLASS_ID=31
      OR CLASS_ID=32
      OR CLASS_ID=33
      OR CLASS_ID=34

```

```

        OR CLASS_ID=35
        OR CLASS_ID=36
        OR CLASS_ID=37
        OR CLASS_ID=38
        OR CLASS_ID=39
        OR CLASS_ID=40
        OR CLASS_ID=41
        OR CLASS_ID=58)
        and lineage=:1)'
using lineage;

--// *****
--// delete city object members
--// *****
execute immediate 'SELECT :1 as message from DUAL'
using 'Deleting city object members...';

--// delete entries from CITYOBJECT_MEMBER
execute immediate 'DELETE FROM CITYOBJECT_MEMBER
        WHERE CITYOBJECT_ID IN
        (SELECT ID FROM CITYOBJECT
        WHERE (CLASS_ID=24
        OR CLASS_ID=25
        OR CLASS_ID=26
        OR CLASS_ID=27
        OR CLASS_ID=28
        OR CLASS_ID=29
        OR CLASS_ID=30
        OR CLASS_ID=31
        OR CLASS_ID=32
        OR CLASS_ID=33
        OR CLASS_ID=34
        OR CLASS_ID=35
        OR CLASS_ID=36
        OR CLASS_ID=37
        OR CLASS_ID=38
        OR CLASS_ID=39
        OR CLASS_ID=40
        OR CLASS_ID=41
        OR CLASS_ID=58)
        and lineage=:1)'

using lineage;

--// *****
--// delete city models
--// *****
execute immediate 'SELECT :1 as message from DUAL'
using 'Deleting city models...';

--// delete entries from APPEARANCE
execute immediate 'DELETE FROM APPEARANCE
        WHERE CITYMODEL_ID IN
        (SELECT ID FROM CITYMODEL
        WHERE (SELECT COUNT(*) FROM CITYOBJECT_MEMBER
        WHERE CITYMODEL_ID = CITYMODEL.ID) = 0)';

--// delete entries from CITYMODEL
execute immediate 'DELETE FROM CITYMODEL
        WHERE (SELECT COUNT(*) FROM CITYOBJECT_MEMBER
        WHERE CITYMODEL_ID = CITYMODEL.ID) = 0';

--// *****
--// delete city objects
--// *****
execute immediate 'SELECT :1 as message from DUAL'
using 'Deleting city objects...';

--// delete entries from CITYOBJECT
execute immediate 'DELETE FROM CITYOBJECT
        WHERE (CLASS_ID=24
        OR CLASS_ID=25
        OR CLASS_ID=26
        OR CLASS_ID=27
        OR CLASS_ID=28
        OR CLASS_ID=29
        OR CLASS_ID=30
        OR CLASS_ID=31
        OR CLASS_ID=32
        OR CLASS_ID=33
        OR CLASS_ID=34
        OR CLASS_ID=35
        OR CLASS_ID=36
        OR CLASS_ID=37
        OR CLASS_ID=38
        OR CLASS_ID=39
        OR CLASS_ID=40
        OR CLASS_ID=41
        OR CLASS_ID=58)
        and lineage=:1'
```

```
using lineage;

--// enable FK-constraints on SURFACE_GEOMETRY
execute immediate 'ALTER TABLE SURFACE_GEOMETRY
    ENABLE
    CONSTRAINT SURFACE_GEOMETRY_FK
    ENABLE
    CONSTRAINT SURFACE_GEOMETRY_FK1';

execute immediate 'SELECT :1 as message from DUAL'
using 'Deletion of buildings complete!';

exception
    when others then
        dbms_output.put_line('delete_lineage: ' || SQLERRM);

end;

END geodb_delete_by_lineage;
/
```

10 Appendix B – Database Setup – Example Session

```
SQL> @create_db
```

```
Please enter a valid SRID (Berlin: 81989002):
Please enter the corresponding SRSName to be used in GML exports
(Berlin: urn:ogc:def:crs,crs:EPSG:6.12:3068,crs:EPSG:6.12:5783):
Shall versioning be enabled (yes/no, default is no): yes
```

```
PL/SQL procedure successfully completed.
```

```
old 3: WHERE SRID=&SRSNO;
new 3: WHERE SRID= 81989002;
old 5: IF (:SRID = &SRSNO) THEN
new 5: IF (:SRID = 81989002) THEN
```

```
PL/SQL procedure successfully completed.
```

```
MC
```

```
-----
CREATE_DB2
```

```
1 row selected.
```

```
Table created.
```

```
Table altered.
```

```
old 1: INSERT INTO DATABASE_SRS (SRID,GML_SRS_NAME) VALUES (&SRSNO,'&GMLSRSNAME')
new 1: INSERT INTO DATABASE_SRS (SRID,GML_SRS_NAME) VALUES (
81989002,'urn:ogc:def:crs,crs:EPSG:6.12:3068,crs:EPSG:6.1
2:5783')
```

```
1 row created.
```

```
Commit complete.
```

```
Table created.
```

```
Table altered.
```

```
[repetitive messages have been omitted for brevity reasons]
```

```
Sequence created.
```

```
[repetitive messages have been omitted for brevity reasons]
```

```
Table altered.
```

```
[repetitive messages have been omitted for brevity reasons]
```

```
Index created.
```

```
[repetitive messages have been omitted for brevity reasons]
```

```
Session altered.
```

```
Session altered.
```

```
PL/SQL procedure successfully completed.
```

```
MC
-----
81989002
```

1 row selected.

Used SRID for spatial indexes: 81989002

1 row deleted.

```
old 7:      MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO)
new 7:      MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)),      81989002)
```

1 row created.

1 row deleted.

```
old 6:      MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)) , &SRSNO)
new 6:      MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)) ,      81989002)
```

1 row created.

[repetitive messages have been omitted for brevity reasons]

1 row deleted.

```
old 5:      MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO)
new 5:      MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)),      81989002)
```

1 row created.

1 row deleted.

```
old 6:      MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO)
new 6:      MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)),      81989002)
```

1 row created.

1 row deleted.

```
old 5:      MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005)), &SRSNO)
new 5:      MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005)),      81989002)
```

1 row created.

1 row deleted.

```
old 6:      MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)), &SRSNO)
new 6:      MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)),      81989002)
```

1 row created.

[repetitive messages have been omitted for brevity reasons]

1 row deleted.

```
old 5:      MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005)), &SRSNO)
new 5:      MDSYS.SDO_DIM_ELEMENT('Y', 0.000, 10000000.000, 0.0005)),      81989002)
```

1 row created.

[repetitive messages have been omitted for brevity reasons]

1 row deleted.

```
old 6:      MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)) , &SRSNO)
new 6:      MDSYS.SDO_DIM_ELEMENT('Z', -1000, 10000, 0.0005)) ,      81989002)
```

1 row created.

[repetitive messages have been omitted for brevity reasons]

Index created.

[repetitive messages have been omitted for brevity reasons]

0 rows deleted.

1 row created.

[repetitive messages have been omitted for brevity reasons]

Commit complete.

Table created.

Table altered.

Procedure created.

0 rows deleted.

1 row created.

PL/SQL procedure successfully completed.

```
old 2: IF ('&VERSIONING'='yes' OR '&VERSIONING'='YES' OR '&VERSIONING'='y' OR
      '&VERSIONING'='Y') THEN
new 2: IF ('yes'='yes' OR 'yes'='YES' OR 'yes'='y' OR 'yes'='Y') THEN
```

PL/SQL procedure successfully completed.

MC2

ENABLEVERSIONING.sql

1 row selected.

MESSAGE

EnableVersioning procedure is working, that takes a while.

1 row selected.

PL/SQL procedure successfully completed.

Call completed.

[repetitive messages have been omitted for brevity reasons]

Procedure created.

[repetitive messages have been omitted for brevity reasons]

MESSAGE

PlanningManager: Creating tables, sequences and stored procedures!

1 row selected.

Commit complete.

[repetitive messages have been omitted for brevity reasons]

1 row deleted.

```
old 7:      &SRSNO)
new 7:      81989002)
```

1 row created.

Table created.

Table altered.

[repetitive messages have been omitted for brevity reasons]

Sequence created.

[repetitive messages have been omitted for brevity reasons]

Index created.

Procedure created.

[repetitive messages have been omitted for brevity reasons]

Commit complete.

No errors.

MESSAGE

PlanningManager: Finished!

1 row selected.

Procedure created.

MESSAGE

Creating packages 'geodb_util', 'geodb_idx', 'geodb_stat', and corresponding types

1 row selected.

Type created.

Package created.

Package body created.

Type created.

Type body created.

Package created.

Package body created.

Package created.

Package body created.

MESSAGE

Packages 'geodb_util', 'geodb_idx', and 'geodb_stat' created

1 row selected.

MESSAGE

Creating matching tool packages 'geodb_match', 'geodb_process_matches', 'geodb_delete_by_lineage', and corresponding types

1 row selected.

drop table match_result

*

ERROR at line 1:

ORA-00942: table or view does not exist

Table created.

```
drop table match_master_aggr_geom
*
```

ERROR at line 1:
ORA-00942: table or view does not exist

Table created.

```
drop table match_cand_aggr_geom
*
```

ERROR at line 1:
ORA-00942: table or view does not exist

Table created.

```
drop table match_allocate_geom
*
```

ERROR at line 1:
ORA-00942: table or view does not exist

Table created.

```
truncate table match_tmp_building
*
```

ERROR at line 1:
ORA-00942: table or view does not exist

```
drop table match_tmp_building
*
```

ERROR at line 1:
ORA-00942: table or view does not exist

Table created.

Package created.

Package body created.

```
truncate table MATCH_RESULT_RELEVANT
*
```

ERROR at line 1:
ORA-00942: table or view does not exist

```
drop table MATCH_RESULT_RELEVANT
*
```

ERROR at line 1:
ORA-00942: table or view does not exist

Table created.

```
drop table COLLECT_GEOM
*
```

ERROR at line 1:
ORA-00942: table or view does not exist

Table created.

```
drop table CONTAINER_IDS
*
```

ERROR at line 1:
ORA-00942: table or view does not exist

Table created.

Package created.

Package body created.

Package created.

Package body created.

MESSAGE

Packages 'geodb_match', 'geodb_process_matches', and 'geodb_delete_by_lineage' created

1 row selected.

No errors.

Commit complete.

MESSAGE

DB creation complete!

1 row selected.

SQL>

11 Appendix C – List of figures and tables

Figure 1: Different types of aggregated geometries (from [Gröger et al., 2008]).....	17
Figure 2: Geometrical-topographical model	18
Figure 3: Appearance model	20
Figure 4: Core Model and thematic top level classes.....	23
Figure 5: Example of buildings consisting of building parts	24
Figure 6: UML diagram of building model.....	25
Figure 7: Boundary surfaces	26
Figure 8: City furniture model	27
Figure 9: UML diagram representing the digital terrain model	28
Figure 10: A relief, consisting of three components and its validity polygons.....	29
Figure 11: Generic CityObject model	30
Figure 12: Landuse model.....	31
Figure 13: LoD2 representation of a transportation complex	31
Figure 14: UML model for transportation complex.....	32
Figure 15: Image illustrates objects of the vegetation model.....	33
Figure 16: Vegetation Model	33
Figure 17: Definition of waterbody attributes.....	34
Figure 18: Waterbody model.....	35
Figure 19: Database schema of the CityGML core elements.....	38
Figure 20: Geometry hierarchy	39
Figure 21: LoD 1 building - closed volume bounded by a CompositeSurface	40
Figure 22: Appearance database schema	44
Figure 23: Simple example explaining texture mapping using texture coordinates	45
Figure 24: Visualisation of a simple building in LoD1 and LoD2.....	46
Figure 25: Images for parameterized textures.....	48
Figure 26: Surface geometries for the building in LoD 2	49
Figure 27: Images for georeferenced textures.....	48
Figure 30: CityFurniture database schema.....	53
Figure 29: LoD2 building with roof overhangs, highlighted in red	51
Figure 30: CityFurniture database schema.....	53
Figure 31: Digital Terrain Model database schema	54
Figure 32: GenericCityObject and generic attributes database schema.....	56
Figure 33: LandUse database schema	58
Figure 34: Orthophoto database schema	59
Figure 35: Transportation database schema	61
Figure 36: Vegetation database schema	62
Figure 37: WaterBody database schema	64
Figure 38: Tables for spatial information and sequences.....	65
Figure 39: Example for database setup	68
Figure 40: CLI help text.....	71
Figure 41: Database connection	73
Figure 42: Graphical User Interface for data import.....	74
Figure 43: Export dialog window.....	77
Figure 44: Import preferences – Continuation	79
Figure 45: Import preferences – ID Handling.....	80
Figure 46: Import preferences – Spatial and normal Indexes	81
Figure 47: Import preferences – Appearance	83
Figure 48: Import preferences – Bounding Box.....	84
Figure 49: Import preferences – XML Validation	85

Figure 50: Import preferences – Resources.....	87
Figure 51: Export preferences – CityGML modules.....	90
Figure 52: Export preferences – Appearance.....	91
Figure 53: Export preferences – Bounding Box.....	92
Figure 54: Export preferences – Resources.....	93
Figure 55: General Preferences – Logging.....	95
Figure 56: General Preferences – Import and Export Path.....	97
Figure 57: General preferences – Language selection.....	98
Figure 58: The overlap of two objects.....	100
Figure 59: The complexity of the building model.....	101
Figure 60: GML surface geometry classes (excerpt).....	102
Figure 61: User interface of the matching tool.....	103
Figure 62: Options for the matching table output length.....	104
Figure 63: Options for handling of GML names from the merge data set.....	105
Figure 64: Options for data cleanup.....	105
Figure 65: Overview of version and history management.....	115
Figure 66: UML diagram comprising classes, attributes and methods for parallel plannings.....	116
Figure 67: Database schema for parallel plannings.....	117
Figure 68: PlanningManager: Interface for the management of plannings.....	125
Figure 69: PlanningManager: Interface for the management of plannings.....	127
Figure 70: The raster import / export tool.....	133
Figure 71: Login mask.....	134

Table 1: Cityobjectgroup tables.....	37
Table 2: Attributes determining aggregation types.....	40
Table 3: Excerpt of table SURFACE_GEOMETRY.....	41
Table 4: Example for table TEXTUREPARAM.....	45
Table 5: Example showing an excerpt of SURFACE_DATA table.....	46
Table 6: Excerpt of table APPEARANCE.....	48
Table 7: Excerpt of table SURFACE_DATA.....	48
Table 8: Table TEXTUREPARAM.....	48
Table 9: Excerpt of table SURFACE_GEOMETRY.....	51
Table 10: Excerpt of table BUILDING.....	52
Table 11: Excerpt of table THEMATIC_SURFACE.....	52
Table 12: Attribute type.....	56
Table 13: Class names.....	57
Table 14: Sequences for tables related to orthophotos.....	60
Table 15: Functions for Planning Manager.....	118