3DIM database implementation test report

Abstract

The 3DIM (3D Integrated Model) is an information model under development which intends to integrate geographic features on the earth surface as well as above and below the earth surface into a common semantic-geometric model. 3DIM is designed to constitute a base model for 3D environments, including geometric features from different domains without including semantics and attributes that are specific for a certain domain. Thus, the model is meant to be application independent. Two database implementation alternatives limited to include the geometry types point, curve, surface and solid are defined. In the first alternative semantics are separated from geometry into two table groups while in the second alternative semantic tables incorporates geometry using data types. This paper describes data preparation, testing of the data loading process and retrieval of data of the two alternative implementations in an Oracle Spatial 11g beta database using FME data processing software and GO Publisher from Snowflake Software. The Oracle SQL Developer is used as database interface. A comparison between the both alternatives is also provided.

Ludvig Emgård, M.Sc
Delft University of Technology / SWECO Position

Content

Input data

The test dataset that was used was created from GIS data from GBKN and laser scan pointclouds from LIDAR data? covering the campus area of the Technical University of Delft, Netherlands. Lacking geological data of Delft, the geological data that was used was gathered from a dataset of an area in Malmö, Sweden. Data describing thematic semantic objects that was not available in these datasets was figured within the datasets by manual 3D modeling performed in 3D studio MAX. More details about GIS data and LIDAR data?

Input data preparation

Earth surface coverage process

Since the earth surface model in 3DIM is defined as a classified coverage the input data to the constraint triangulation that was used had to consist of a fully partitioned coverage, meaning that the areas are not allowed to overlap and one part of the surface can only be occupied by one feature. The GIS data retrieved from GBKN was based on curve (polyline) features that were fairly topologically connected. Using the FME software all line features within a decided test area was input into the FME PolygonBuilder transformer. The DWG file containing the GIS data was analyzed and split into several files depending on semantic or geometric properties. A set of curve layers including buildings, roads, walls, fences, tree lines and water boundaries was chosen to be included in the coverage generation. Arc features were stroked to line features using 6 interpolated points in the line representing the arc. All lines were snapped and intersected against each other before entering the polygon build process. To avoid donut polygons all polygons containing holes were sliced into pieces by a 25x25 meter grid. For polygons that still contained holes after subdivision the holes were removed.

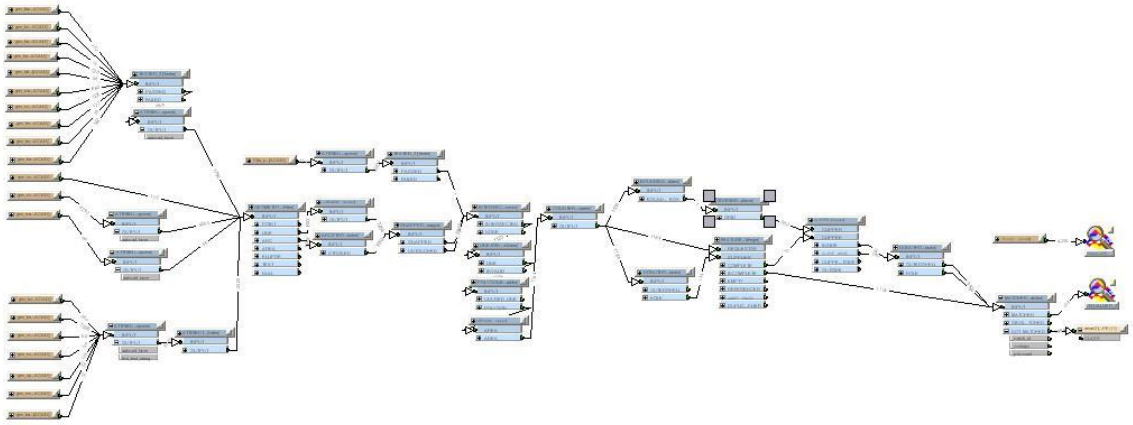


Figure X. FME workbench creating area coverage from line features.

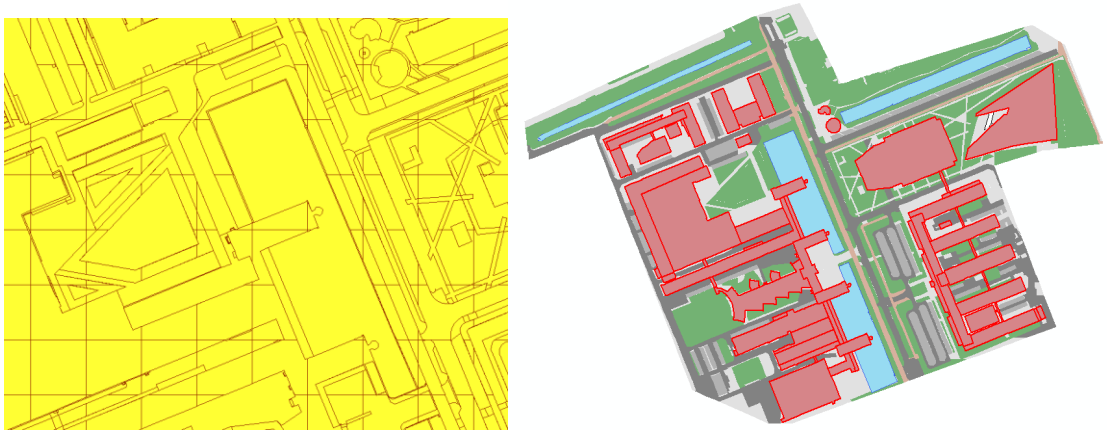
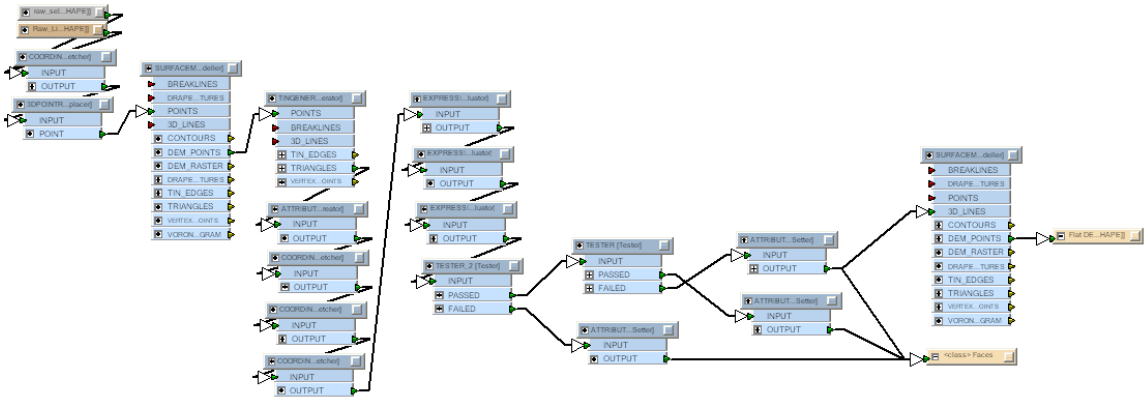


Figure X and X. The area coverage and the classified areas.

The coverage was written to a shape file to enable import in ArcGIS. Using ArcGIS all areas were classified into classes building, water, grass or road. The road polygons were further classified into roads for cycling, cars, parking and other. The class *other* also included other areas not used particularly for transportation. The building polygons were divided into pieces according to the roof structures since different parts of a building footprint can have several elevation values in an extrusion model of the footprint. The red outline in figure X marks the different pieces of the buildings. The mean value of the elevation observations from the point cloud data was calculated for each building roof element. The mean value was attached as an attribute to each building polygon. The area, covering approximately 450x450 meters including 1665 polygons was manually classified in six hours.

Elevation data resampling process



The lidar elevation point cloud was filtered and resampled before the earth surface model was created. The filtering process was based on following methods:

1. The original lidar elevation points were resampled into a 1x1m raster reducing the amount of points from 1376851 to 324960 using a surface tolerance of 5.0 in the FME surfacemodeller transformer.
2. A TIN was created from the 324960 points
3. The slope of each edge in each triangle was calculated
4. If any of the edges of each triangle was sloped more than 0.05 radians this triangle was considered sloped.
5. Sloped triangles were removed from further calculations
6. Since the earth surface ground level in this area is reasonably flat and the ground level is slightly below sea level all triangles that had at least one vertex below 0 was considered non ground triangles.
7. All flat triangles that were considered ground triangles were input in the surface modeler transformer as 3D features.
8. Once again with the surface tolerance of 5.0 the surface modeler did output a 4 x 4m raster containing 19920 elevation values considered earth surface ground values.

Creation of terrain intersection objects

Within the 2D GIS data, points and curves were selected to be used as 3DIM terrain intersection objects in the test dataset. Trees and streetlights were represented by points and walls were represented by lines. In addition tree rows and streetlight rows were created from curve geometries.

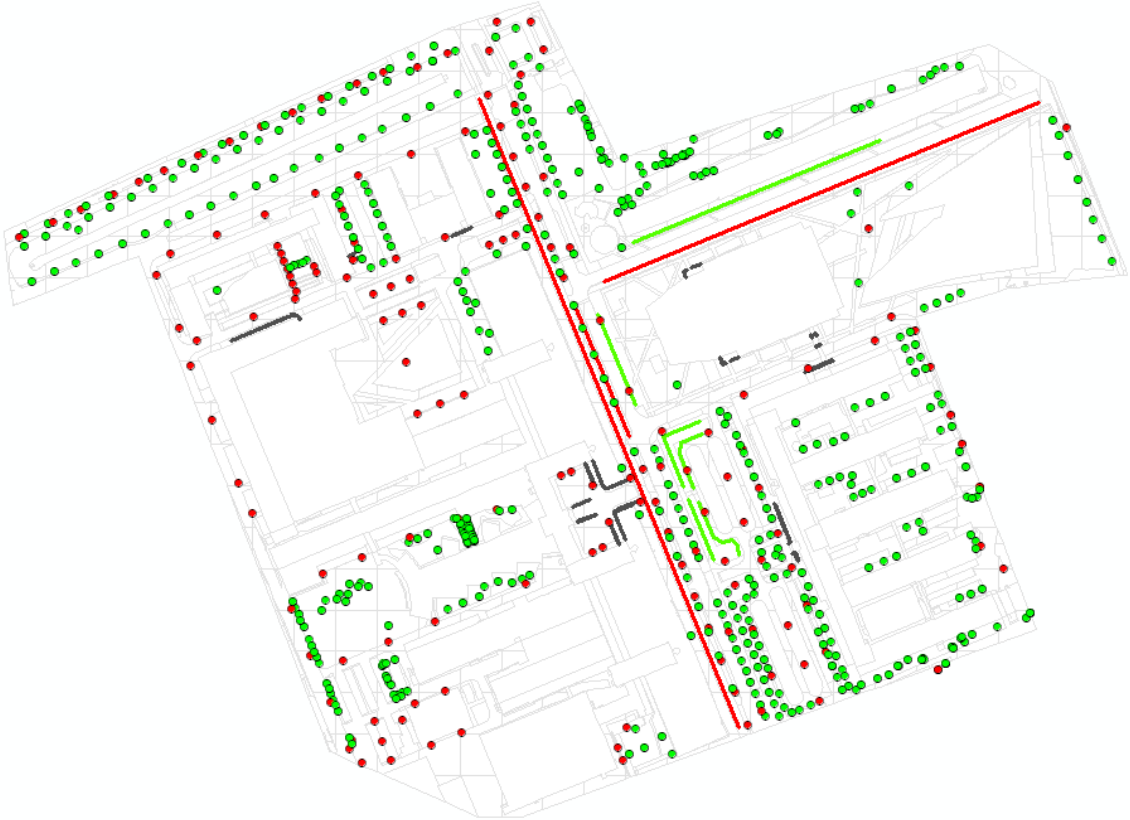


Figure X. Trees in green points, Streetlights in red points. Walls in black lines, tree rows in green lines and streetlight rows in red lines.

An attribute was created for each tree and streetlight row indicating the number of trees or streetlights distributed on an equal distance along the line. The point features and line features were stored in separate files.

Feature	**Amount**
Trees	1318
Streetlights	158
Walls	196
Tree rows	5
Street light rows	3

Constraint earth surface modeling and building extrusion

The elevation data, the coverage polygons and the terrain intersection points and curves where processed together in another FME workbench. All polygon, line and terrain intersection objects where input to a surfacedraper transformer as drape features. The elevation points where input in the same transformer as point features. Hence, an elevation model was created from the points and each vertex of all drape features received an interpolated elevation value. The tolerance value of the surfacedraper was set to 0 meaning that all input geometries were considered. After gaining z-values the terrain intersection objects were output and not treated further. The TIN model was created using all elevation values from the 4x4 DEM and the terrain intersection points. All curve and polygon features were input as constraining 3D lines. The TIN model was then classified according to the classes in the coverage polygon file using the clipper transformer to match triangles against polygons. After this operation all triangles were classified and output in separate groups according to their classifications.

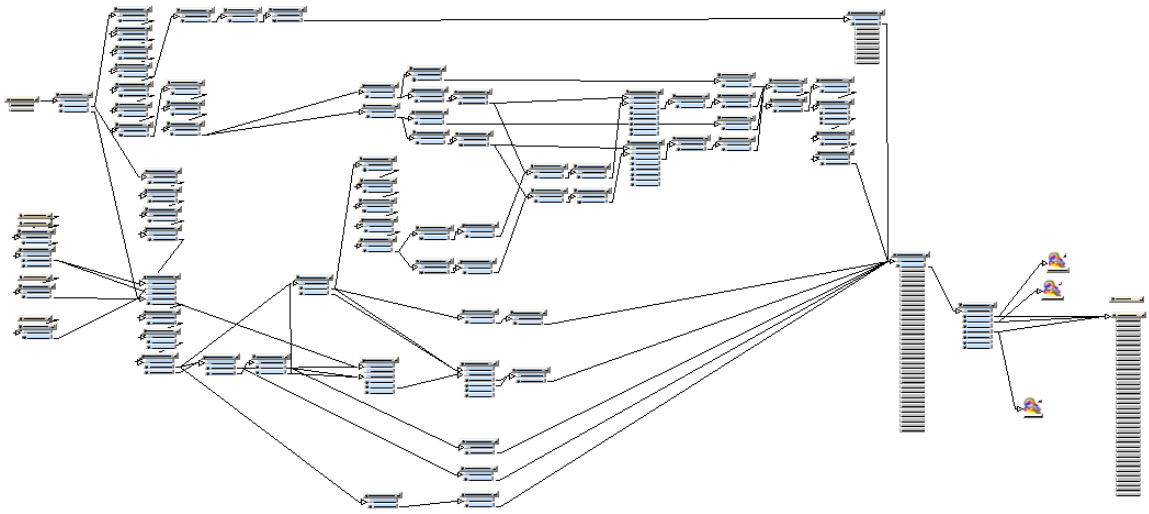


Figure X. FME workbench for surface modeling and building extrusion

The building polygons from the coverage polygon file contained an elevation value of each building element that was used for the extrusion. All vertices of each polygon were set to the elevation height and extruded downwards to the draped building footprint (terrain intersection curve) as follows:

1. The elevated polygons were oriented according to the right hand rule (counter clockwise order when considering the polygon from above.
2. An ID was given to each polygon in an attribute _polygoncount
3. The coordinates of each polygon were counted in the attribute _coordinatecount
4. Each polygon was cloned in the same number of instances as the number of vertices for the polygon (_coordinatecount)
5. By aggregating and deaggregating all polygons with the same _polygoncount attribute a vertex index (vertex1index) was assigned to each polygon according to the _coordinatecount attribute.
6. An additional attribute (vertex2index) was assigned to each polygon given the value of vertex1index+1.
7. The coordinates of the vertices of vertex1index and vertex2index were fetched and assigned in attributes _x and _y for each polygon.
8. The geometry of each polygon feature is replaced by two points. One point with coordinates of vertex1 and one point with the coordinates of vertex 2. The first point is assigned the z-value of the polygon and the second point is assigned the z-value of the draped footprint polygon. In this way two instances of all the coordinates of each planar rectangle that is constituting the walls are created
9. The 3D point features were grouped and sorted according to their original polygon belonging and their vertex indices.
10. The points were connected by polyline features and four line segments were created for each group.
11. The line segments were closed into polygon features. Each polygon feature kept the _polygoncount attribute and thus the belonging to the original object is preserved.

The created roof and wall segments were finally output to an output file. To be able to visualize the data after the processing and extrusion all polygons with more than three vertices was converted to mesh features in Microstation XM V8. The resulting data was visualized in 3D-

studio. Because of the classified earth surface the different ground types could be easily textured with a repeated texture for each class (see figure X)

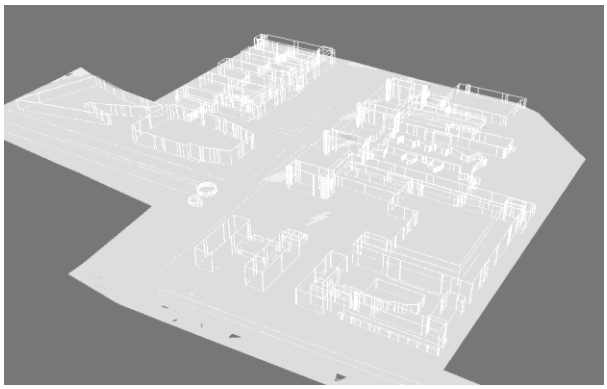
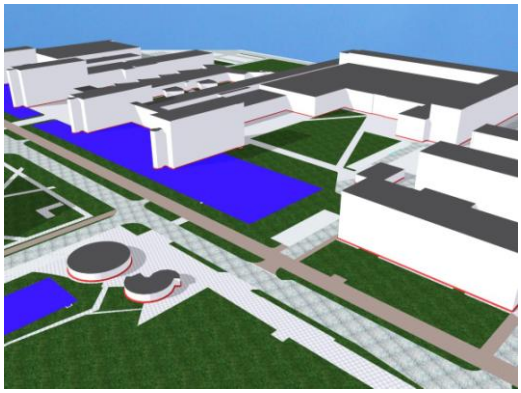



Figure X and X. Output from FME process and a rendered textured view of the dataset.

Preparation of geological data

The geological dataset was retrieved as a shape file with five geological layers stored as polygon features containing a z-value for each vertex and an attribute indicating the geological layer. The dataset was simply split into one file for each layer to simplify input in the database. The geologic layers were placed underneath the earth surface model (see figure X).

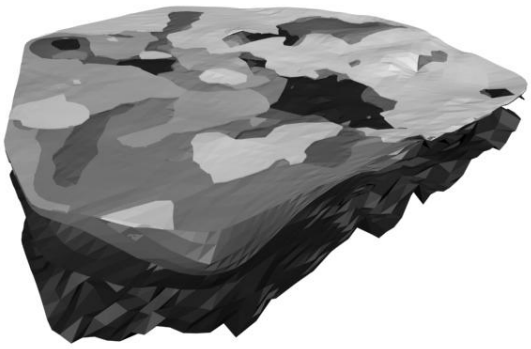
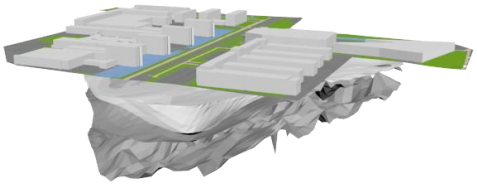



Figure X and X. Geological layers and placement under the earth surface model

Simulation of unavailable data

Since no data was available for the 3DIM classes *BelowSurfaceSpace*, *ConstructionWork*, *Utilities* or *Water* (in body shape) one geometric instance of each class was created for testing in the database model. A small basement, a stone statue and the body of one of the water masses was modeled and two pipes were modeled beneath the main road.

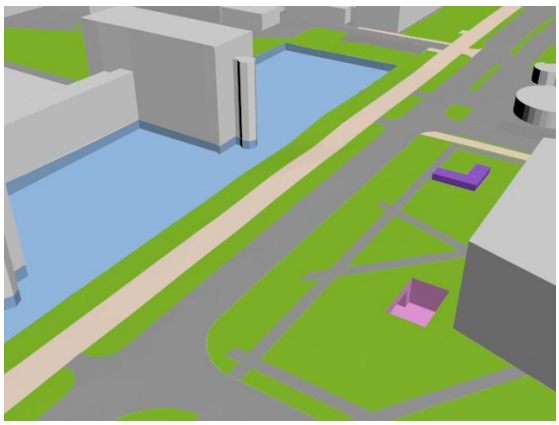
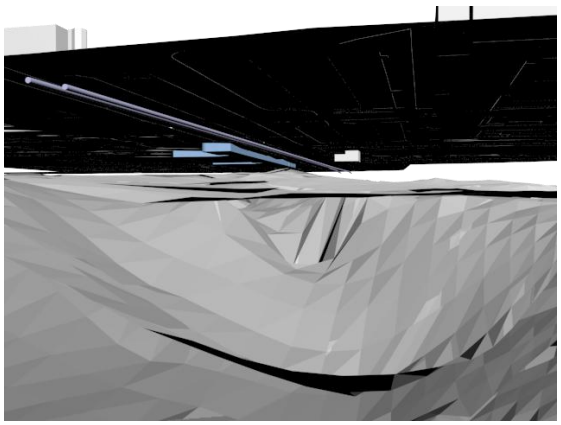



Figure X and X. Manually modeled features

Symbolic data

Two symbols are used in the dataset; a tree of class *Vegetation* and a streetlight of class *CityFurniture.*

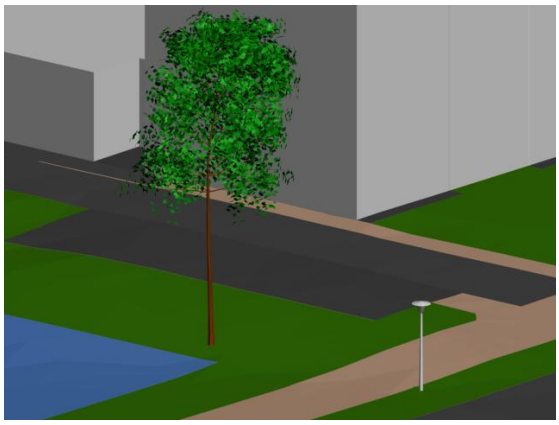


Figure X. Tree and streetlight symbols

Database import

Table creation

The tables of the both database implementation alternatives were created using the Enterprise Architect software and exported directly to SQL scripts. The SQL scripts were run by the database manager software SQL Developer into an Oracle Spatial 11g beta database. Tables, keys and constraint were created by the database while user defined data types export were not supported. Thus, the two data types used in alternative II was created by manually coded SQL statements. In addition, the insertion in the user_sdo_geom_metadata table was also manually coded for the tables including sdo_geometry columns. The metadata insertions were necessary for defining the dimensions of geometry in the geometry column. Without metadata information the FME software that was used to input data could not resolve the number of dimensions of the geometry column and by default set the dimensions to two. Alternative I contained 17 tables and

5 metadata insertions while alternative II contained 14 tables and 14 metadata insertions. The SQL-scripts for the both implementation alternatives is included in Appendix1.

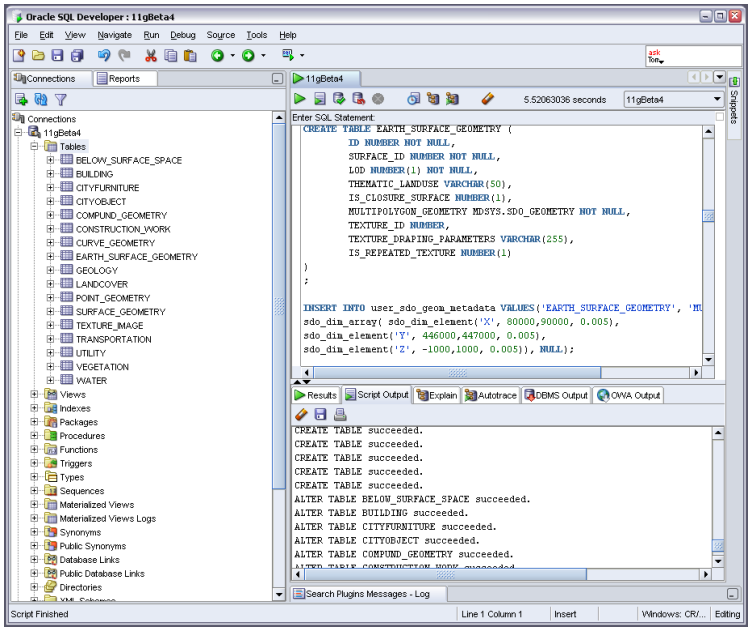


Figure X. Tables and scripts in implementation alternative I

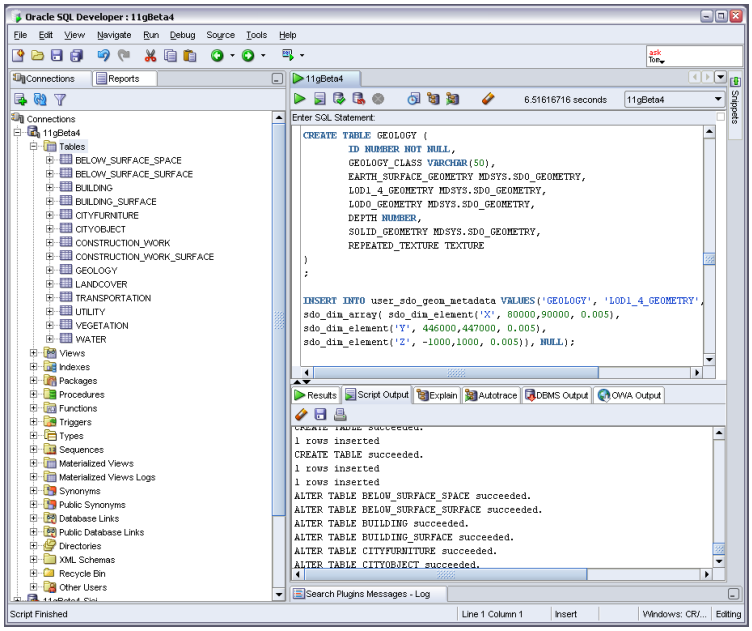


Figure X. Tables and scripts in implementation alternative II

Data import

The data was imported using FME Professional Oracle Suite with possibility to read and write features to an Oracle Spatial database. For alternative I four FME workbenches were created. The reason for dividing the loading process in different workbenches was that the geometry tables were used for several semantic classes for example SURFACE_GEOMETRY. Merging all workbenches would make the diagrams more complex to follow.

Alternative I

AltI - DATABASE_FILL_ABOVE_BELOW.fmw	Buildings BelowSurfaceSpace Water ConstructionWork
AltI - DATABASE_FILL_EARTH_SURFACE.fmw	Transportation Landcover
AltI - DATABASE_FILL_GEO&UTIL.fmw	Geology Utility
AltI - DATABASE_FILL_TIC&TIP.fmw	Vegetation CityFurniture

Figure X Workbench subdivision and thematic semantic classes treated in each workbench for implementation alternative I.

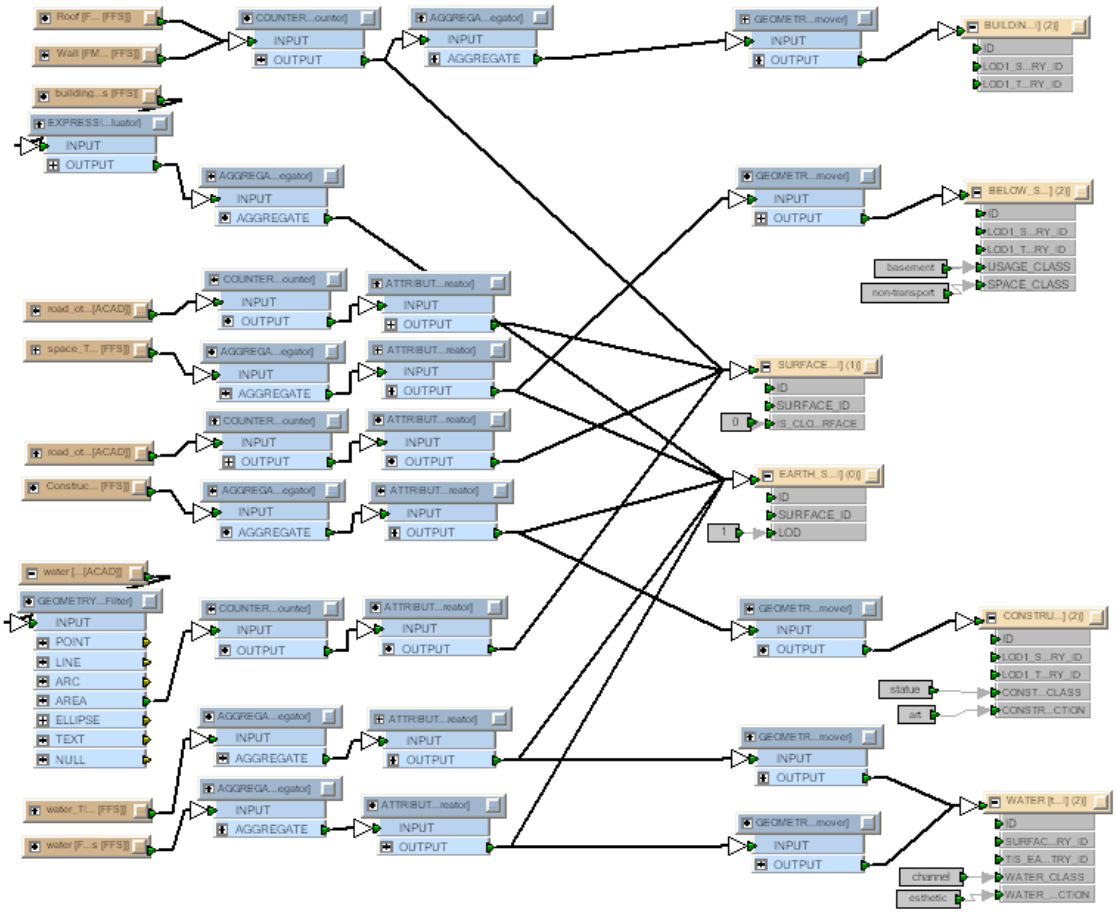


Figure X. Workbench for input of above and below surface features connected to *TerrainIntersectionSurfaces*.

Above and below surface objects connected to *TerrainIntersectionSurfaces* was grouped in one workbench. As can be observed in figure X all input data source groups are routed to the thematic semantic table (e.g. building or water) and the surface geometry table as well as the earth surface geometry table. With the foreign key constraint between the geometric and the semantic table the geometric feature must always be written before the semantic feature. By placing the geometry remover before each semantic destination data table the geometry was always written first and the constraint was not violated. Attribute values not available in the source data was simulated using constant values as can be seen next to the destination data tables. All features in the workbench consist of polygon geometries forming a boundary of the features. In the surface geometry table polygons were input as single polygons in the polygon geometry sdo_geometry column. The earth surface part of the geometry was aggregated to multipolygon geometry before it was input in the multipolygon sdo_geometry column of the earth surface geometry table.

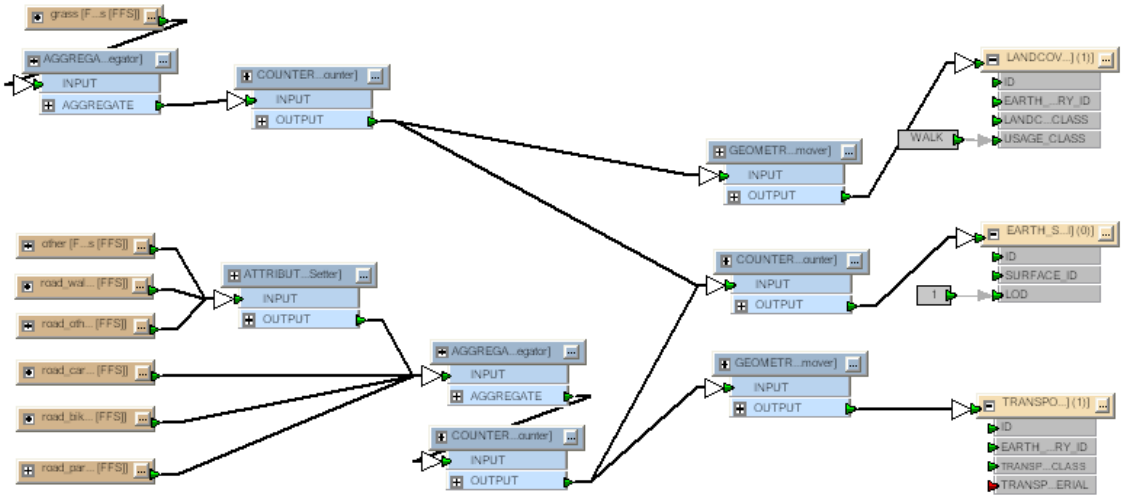


Figure X. Workbench for input of earth surface features

Earth surface objects were aggregated to multipolygon features and input in the semantic table (landcover and transportation) as well as in the earth surface geometry table. The name of the source data file was written as the class attribute in the semantic tables. For example the file road_car.ffs was written as road_car in the transportation class attribute of the transportation table.

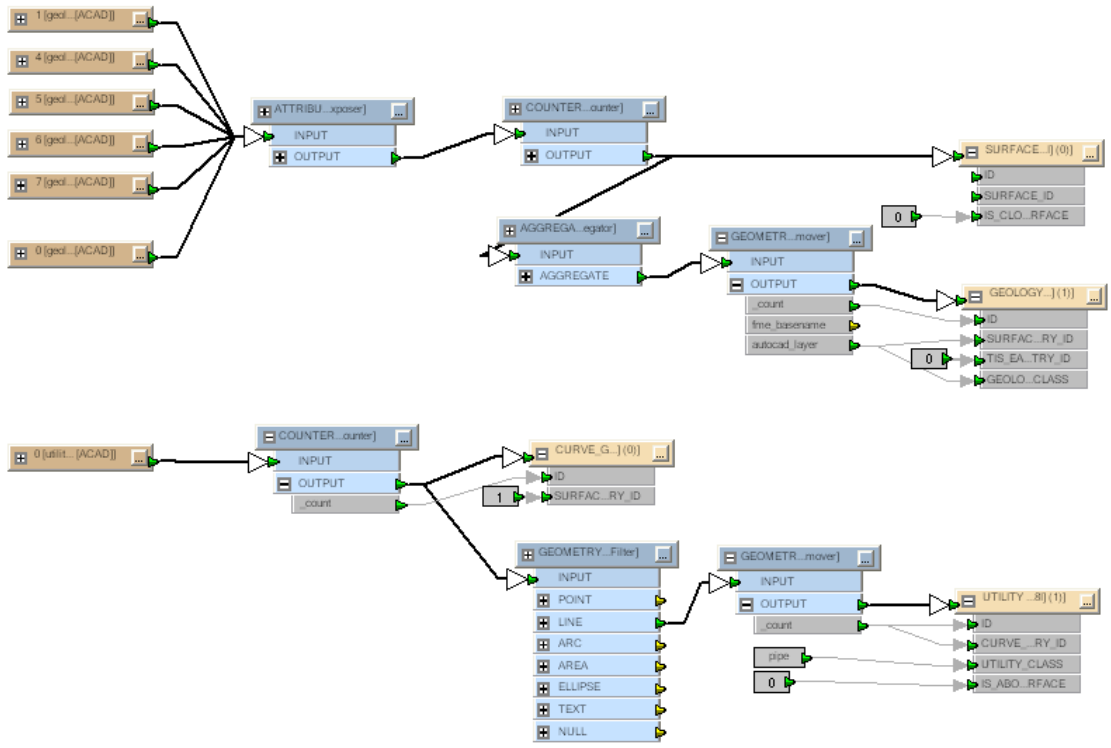


Figure X. Workbench for input of geology and utility features

Geologic features were imported in a manner similar to buildings even though the geologic features in the example dataset were not intersecting the earth surface and thus were not input into the earth surface geometry table. As can be observed in figure X the utilities are input in the curve geometry table as well as in the utility table (without geometry).

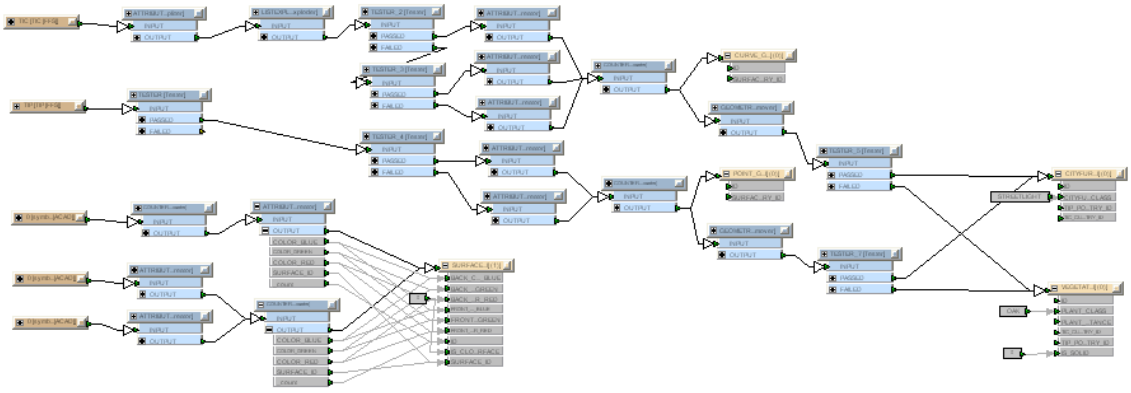


Figure X. Workbench for input of vegetation, city furniture

Since both the vegetation class and the cityfurniture class in the dataset contain both curve (TIC) and point (TIP) geometries the routing of these features to the proper data table was slightly more complex. Points were input in one file and curves in another. With help from the tester transformer (routing depending on attribute values) the data could be routed to the proper

destination table. The tree and streetlight symbols were simply written to the surface geometry table.

Alternative II

AltII - DATABASE_FILL_ABOVE_BELOW.fmw	Buildings BelowSurfaceSpace Water ConstructionWork
AltII - DATABASE_FILL_ABOVE_BELOW_WATER.fmw	Water
AltII - DATABASE_FILL_EARTH_SURFACE.fmw	Transportation Landcover
AltII - DATABASE_FILL_GEO&UTIL.fmw	Geology Utility
AltII - DATABASE_FILL_TIC&TIP.fmw	Vegetation CityFurniture

Figure X Workbench subdivision and thematic semantic classes treated in each workbench for implementation alternative II.

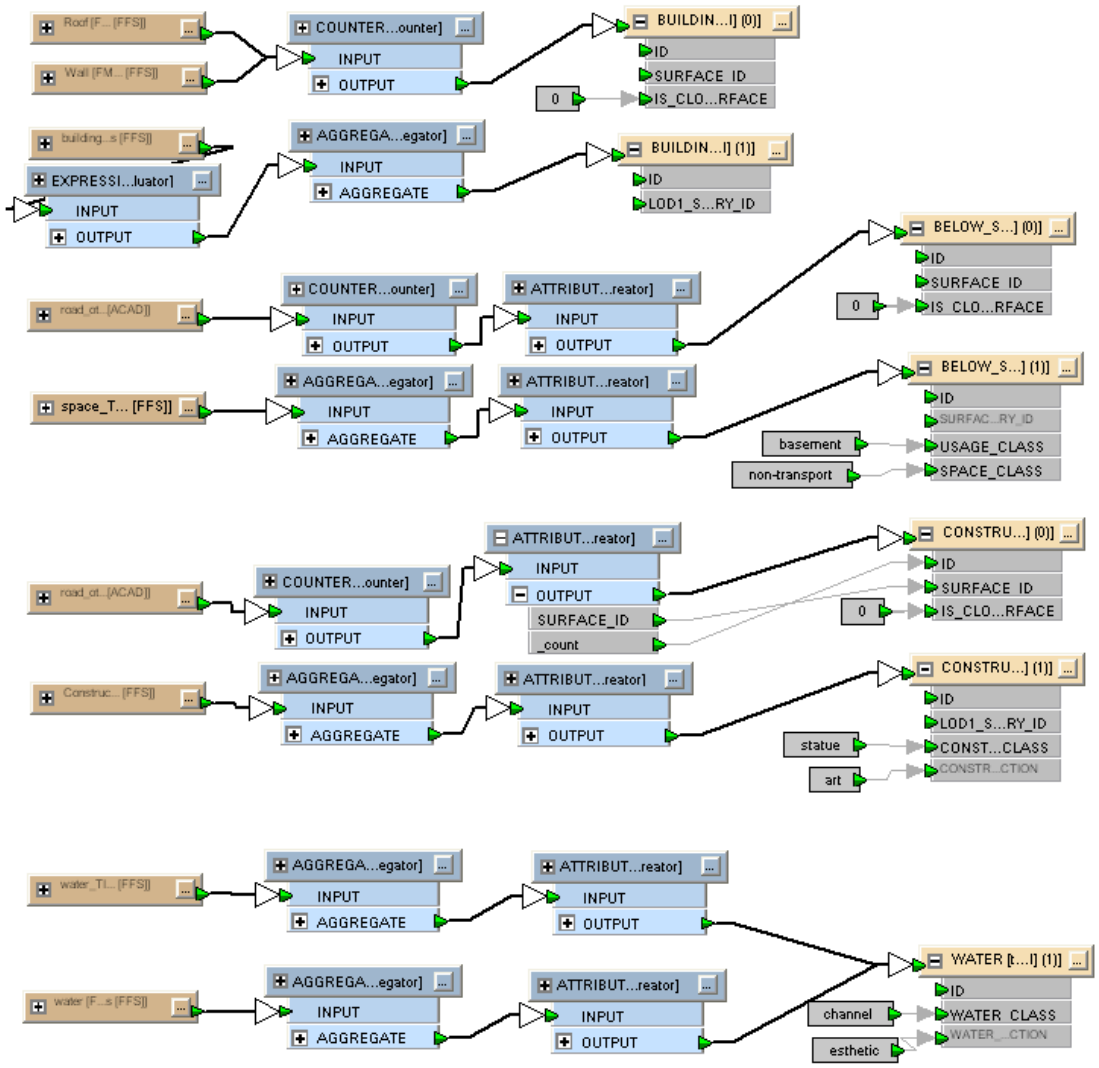


Figure X. Workbench for input of above and below surface features connected to *TerrainIntersectionSurfaces*.

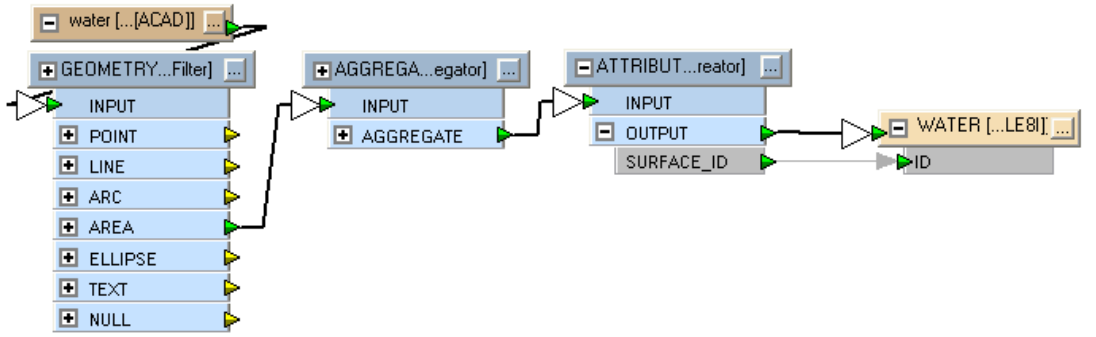


Figure X. Workbench for input of parts of the earth surface geometry part of the water class.

In similarity with the above and below surface feature workbench in alternative I all the polygon geometries are aggregated and written with multipolygon geometries. Since here no division exists between geometric and semantic table the writing process is straighter forward. Due to the fact that the water table had to be accessed two times in different geometry columns the loading process of water was separated into two workbenches; one water feature may have two geometries. In the second workbench (figure X) the water feature is updated with its second geometry using the same id to identify the row.

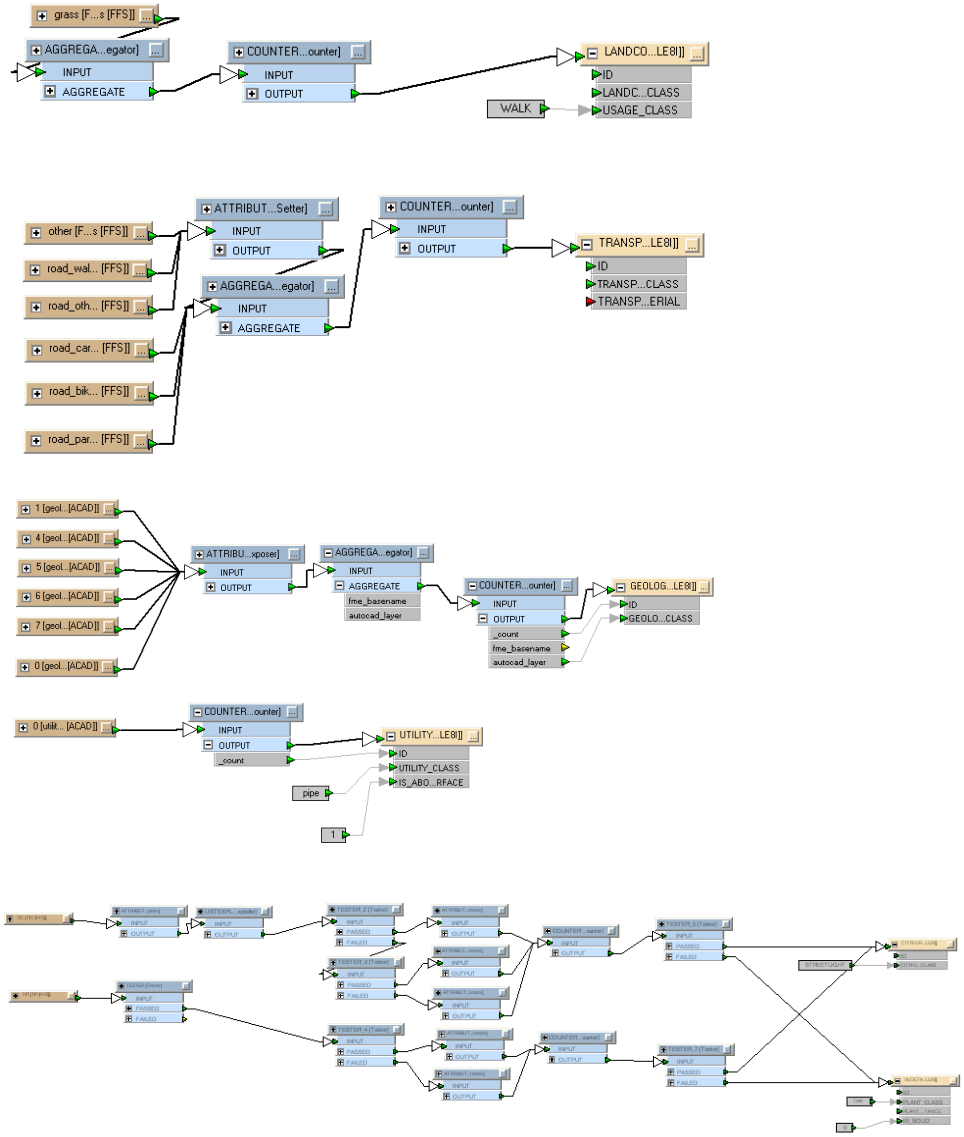


Figure X. Workbenches for input of earth surface features, geology, utility, vegetation and city furniture.

The three remaining workbenches in figure X are similar to the workbenches for the same classes in alternative one except for the geometric table that are removed in alternative II. The symbol creation is not handled by FME because in alternative II symbols are represented by BLOBs.

Data import verification

To verify that the test dataset was properly imported to the database a X3D browser was used, connected to a web service for instant creation of X3C files from the content of a database query (de Vries). By selecting different tables with an SQL statement data from the database could be inspected. No data quality losses or inconsistencies were identified.

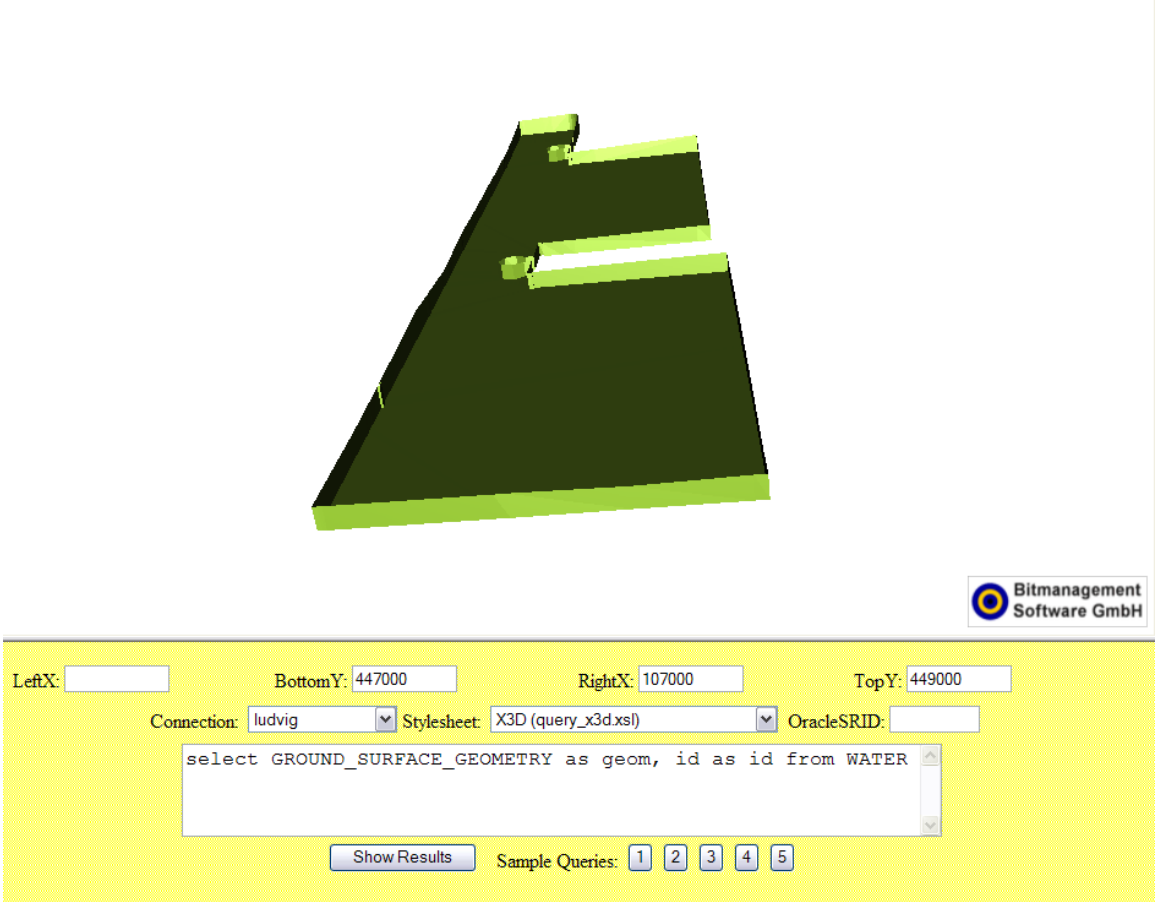


Figure X. A water body geometry verified in the X3D viewer application.

CityGML export

All features classes developed in the 3DIM conceptual model do not exist in the CityGML. For example the construction work class and the subsurface objects are feature classes belonging only to 3DIM. The features classes that exist in both CityGML and 3DIM are Building, Vegetation, City furniture, Transportation, Water and the Earth surface. The data belonging to these classes was exported to CityGML files using the GO Publisher software from Snowflake Software. Go Publisher connects to the spatial database and creates CityGML files based on a mapping between the CityGML schema and the database tables. The mapping from the two alternatives to CityGML is described further.

Creating views

To enable direct mapping from the database implementations, views were created on top of some of the database tables. For example, in many cases two tables are used to describe one semantic feature; the building features class is in both alternatives composed by a combination of two tables. The joins, aggregations, unions and filters that were required could be created directly in the GO Publisher software, but to also enable other software to use the operations simply native views in Oracle Spatial were chosen as method. A complete description of the views can be found in appendix B.

Views in implementation alternative I

```
create or replace view VIEW_BUILDING_LOD1 as (
SELECT a.ID, a.NAME, b.GEOM AS LOD_1_SURFACE_GEOMETRY,
c.MULTIPOLYGON_GEOMETRY AS LOD_1_EARTH_SURFACE_GEOMETRY
FROM BUILDING a, VIEW_BUILDING_AGGRE_SURFACE b,
EARTH_SURFACE_GEOMETRY c
WHERE a.LOD1_TIS_E_S_GEOMETRY_ID = c.SURFACE_ID AND
a.LOD1_SURFACE_GEOMETRY_ID = b.SURFACE_ID
);

create or replace view VIEW_BUILDING_AGGRE_SURFACE as (
SELECT SURFACE_ID as SURFACE_ID, SDO_AGGR_UNION(
SDOAGGRTYPE(d.POLYGON_GEOMETRY, 0.005)) AS GEOM
FROM SURFACE_GEOMETRY d
WHERE SURFACE_ID IN (SELECT LOD1_SURFACE_GEOMETRY_ID FROM
BUILDING)
GROUP BY SURFACE_ID
);
```

Figure X. Examples of building views in implementation alternative I

In implementation alternative I the building and water feature classes are stored using a semantic table (BUILDING), the EARTH SURFACE GEOMETRY table and the SURFACE GEOMETRY table. The example dataset includes only buildings in LOD1 and thus no texture or color properties are added to each individual polygon composing the building. Therefore the single polygons stored in the SURFACE GEOMETRY table were aggregated to a multipolygon feature using the SDO_AGGR_UNION function in Oracle Spatial (see figure X). When joining the semantic table BUILDING with the aggregated view (VIEW_BUILDING_AGGRE_SURFACE) and the EARTH SURFACE GEOMETRY table the final view (VIEW_BUILDING_LOD1) consisted of one row for each building with two multipolygon geometries. A similar operation was applied to the WATER table. The cityfurniture features of point geometry type (streetlights) were managed in the VIEW_POINT_CITY_FURNITURE. The semantic table (CITYFURNTIURE) was joined with the geometric table (POINT_GEOMETRY) to retrieve all cityfurniture features of type point. A similar view was created for the VEGETATION table joining the POINT GEOMETRY table (trees). The LANDCOVER and TRANSPORTATION tables where both joined with the EARTH SURFACE GEOMETRY table so that CityGML transportation and vegetation (surface) features could be created. No view was created for the earth surface model since the geometries were already stored in one table (EARTH SURFACE GEOMETRY).

Views in implementation alternative II

```
create or replace view view_solitary_vegetation as (
SELECT * FROM VEGETATION c WHERE c.LOD0_GEOMETRY.Get_GType() ='1');

create or replace view view_point_city_furniture as (
SELECT * FROM CITYFURNITURE c WHERE c.LOD0_GEOMETRY.Get_GType()
='1');
```

Figure X. Examples of vegetation and cityfurniture views in implementation alternative II

In alternative II the building table structure is very similar to the one in alternative II with the only difference that the surface geometries are in this case stored in a specific BUILDING SURFACE table instead of the more general SURFACE GEOMETRY table. Therefore the aggregated view and the join of tables where created in a similar manner. Since the earth surface elements are in alternative II stored within the semantic building table the join was in this case slightly simpler since only two tables were joined. On the other tables no joined were performed because of the inclusion of geometry into the semantic tables. The VEGETATION and CITYFURNITURE tables were merely filtered according to the geometry type of existing geometries (see figure X) where point features are filtered out. The earth surface created a more complex problem since the pieces of the triangulated surface is scattered in the semantic tables. Therefore a UNION operation was deployed between the tables to create a view containing the geometry of the entire earth surface model. No views were created for the TRANSPORTATION and WATER tables.

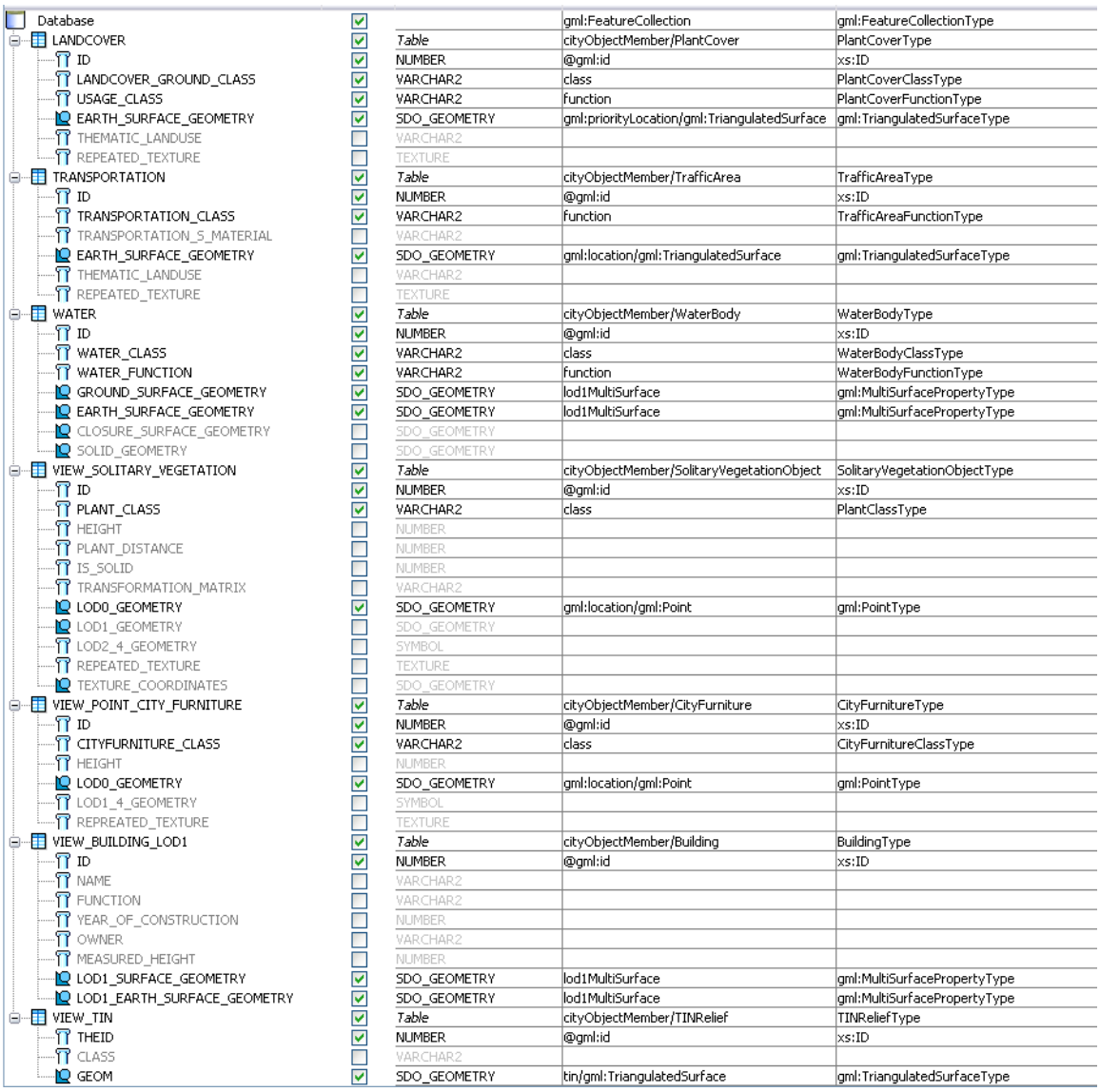
3DIM to CityGML mapping

With the created views the mapping to CityGML could be performed rather straight forward using the Go Publisher software.

Name	Enabled	DB type or const value	XML path	Type in XML
Database	☑		gml:FeatureCollection	gml:FeatureCollectionType
EARTH_SURFACE_GEOMETRY	☑	*Table*	cityObjectMember/TINRelief	TINReliefType
ID	☑	NUMBER	@gml:id	xs:ID
SURFACE_ID	☐	NUMBER		
LOD	☐	NUMBER		
THEMATIC_LANDUSE	☐	VARCHAR2		
IS_CLOSURE_SURFACE	☐	NUMBER		
MULTIPOLYGON_GEOMETRY	☑	SDO_GEOMETRY	tin/gml:TriangulatedSurface	gml:TriangulatedSurfaceType
TEXTURE_ID	☐	NUMBER		
TEXTURE_DRAPING_PARAMETERS	☐	VARCHAR2		
IS_REPEATED_TEXTURE	☐	NUMBER		
VIEW_BUILDING_LOD1	☑	*Table*	cityObjectMember/Building	BuildingType
ID	☑	NUMBER	@gml:id	xs:ID
NAME	☐	VARCHAR2		
LOD_1_SURFACE_GEOMETRY	☐	SDO_GEOMETRY	lod1MultiSurface	gml:MultiSurfacePropertyType
LOD_1_EARTH_SURFACE_GEOMETRY	☑	SDO_GEOMETRY	lod1MultiSurface	gml:MultiSurfacePropertyType
VIEW_LANDCOVER	☑	*Table*	cityObjectMember/PlantCover	PlantCoverType
ID	☑	NUMBER	@gml:id	xs:ID
LANDCOVER_GROUND_CLASS	☑	VARCHAR2	class	PlantCoverClassType
USAGE_CLASS	☑	VARCHAR2	function	PlantCoverFunctionType
MULTIPOLYGON_GEOMETRY	☑	SDO_GEOMETRY	lod1MultiSurface	gml:MultiSurfacePropertyType
VIEW_POINT_CITY_FURNITURE	☑	*Table*	cityObjectMember/CityFurniture	CityFurnitureType
ID	☑	NUMBER	@gml:id	xs:ID
CITYFURNITURE_CLASS	☑	VARCHAR2	class	CityFurnitureClassType
TIP_POINT_GEOMETRY_ID	☐	NUMBER		
POINT_GEOMETRY	☑	SDO_GEOMETRY	lod1Geometry/gml:Point	gml:PointType
VIEW_POINT_VEGETATION	☑	*Table*	cityObjectMember/SolitaryVegetationObject	SolitaryVegetationObjectType
ID	☑	NUMBER	@gml:id	xs:ID
PLANT_CLASS	☑	VARCHAR2	class	PlantClassType
TIP_POINT_GEOMETRY_ID	☐	NUMBER		
POINT_GEOMETRY	☑	SDO_GEOMETRY	lod1Geometry/gml:Point	gml:PointType
VIEW_TRANSPORTATION	☑	*Table*	cityObjectMember/TrafficArea	TrafficAreaType
ID	☑	NUMBER	@gml:id	xs:ID
TRANSPORTATION_CLASS	☑	VARCHAR2	function	TrafficAreaFunctionType
TRANSPORTATION_S_MATERIAL	☐	VARCHAR2		
MULTIPOLYGON_GEOMETRY	☑	SDO_GEOMETRY	lod2MultiSurface	gml:MultiSurfacePropertyType
VIEW_WATER	☑	*Table*	cityObjectMember/WaterBody	WaterBodyType
ID	☑	NUMBER	@gml:id	xs:ID
WATER_CLASS	☑	VARCHAR2	class	WaterBodyClassType
WATER_FUNCTION	☑	VARCHAR2	function	WaterBodyFunctionType
WATER_GROUND_SURFACE_GEOMETRY	☑	SDO_GEOMETRY	lod1MultiSurface	gml:MultiSurfacePropertyType
WATER_EARTH_SURFACE_GEOMETRY	☑	SDO_GEOMETRY	lod1MultiSurface	gml:MultiSurfacePropertyType

Figure X. Mapping from tables and Views to CityGML using GO Publisher (alternative I)

The EARTH SURFACE GEOMETRY table was mapped to the TINrelief object in CityGML using the gml: TriangulatedSurface to store the multipolygons in the database table. The building view and the water view were mapped to the CityGML classes using two lod1MultiSurface objects representing the geometry of each building and water body. One for the walls and roof and one for the terrain intersection surface (ground floor) of the building. (is this allowed in CityGML???). The landcover class (in this case grass) was mapped to the vegetation object plantcover in CityGML and the transportation class was mapped to Traffic area. Vegetation and cityfurniture was simply mapped to the CityFurniture and SolitaryVegetationObject using gml: point as geometry type.



Database			gml:FeatureCollection	gml:FeatureCollectionType
LANDCOVER		*Table*	cityObjectMember/PlantCover	PlantCoverType
ID		NUMBER	@gml:id	xs:ID
LANDCOVER_GROUND_CLASS		VARCHAR2	class	PlantCoverClassType
USAGE_CLASS		VARCHAR2	function	PlantCoverFunctionType
EARTH_SURFACE_GEOMETRY		SDO_GEOMETRY	gml:priorityLocation/gml:TriangulatedSurface	gml:TriangulatedSurfaceType
THEMATIC_LANDUSE		VARCHAR2		
REPEATED_TEXTURE		TEXTURE		
TRANSPORTATION		*Table*	cityObjectMember/TrafficArea	TrafficAreaType
ID		NUMBER	@gml:id	xs:ID
TRANSPORTATION_CLASS		VARCHAR2	function	TrafficAreaFunctionType
TRANSPORTATION_S_MATERIAL		VARCHAR2		
EARTH_SURFACE_GEOMETRY		SDO_GEOMETRY	gml:location/gml:TriangulatedSurface	gml:TriangulatedSurfaceType
THEMATIC_LANDUSE		VARCHAR2		
REPEATED_TEXTURE		TEXTURE		
WATER		*Table*	cityObjectMember/WaterBody	WaterBodyType
ID		NUMBER	@gml:id	xs:ID
WATER_CLASS		VARCHAR2	class	WaterBodyClassType
WATER_FUNCTION		VARCHAR2	function	WaterBodyFunctionType
GROUND_SURFACE_GEOMETRY		SDO_GEOMETRY	lod1MultiSurface	gml:MultiSurfacePropertyType
EARTH_SURFACE_GEOMETRY		SDO_GEOMETRY	lod1MultiSurface	gml:MultiSurfacePropertyType
CLOSURE_SURFACE_GEOMETRY		SDO_GEOMETRY		
SOLID_GEOMETRY		SDO_GEOMETRY		
VIEW_SOLITARY_VEGETATION		*Table*	cityObjectMember/SolitaryVegetationObject	SolitaryVegetationObjectType
ID		NUMBER	@gml:id	xs:ID
PLANT_CLASS		VARCHAR2	class	PlantClassType
HEIGHT		NUMBER		
PLANT_DISTANCE		NUMBER		
IS_SOLID		NUMBER		
TRANSFORMATION_MATRIX		VARCHAR2		
LOD0_GEOMETRY		SDO_GEOMETRY	gml:location/gml:Point	gml:PointType
LOD1_GEOMETRY		SDO_GEOMETRY		
LOD2_4_GEOMETRY		SYMBOL		
REPEATED_TEXTURE		TEXTURE		
TEXTURE_COORDINATES		SDO_GEOMETRY		
VIEW_POINT_CITY_FURNITURE		*Table*	cityObjectMember/CityFurniture	CityFurnitureType
ID		NUMBER	@gml:id	xs:ID
CITYFURNITURE_CLASS		VARCHAR2	class	CityFurnitureClassType
HEIGHT		NUMBER		
LOD0_GEOMETRY		SDO_GEOMETRY	gml:location/gml:Point	gml:PointType
LOD1_4_GEOMETRY		SYMBOL		
REPREATED_TEXTURE		TEXTURE		
VIEW_BUILDING_LOD1		*Table*	cityObjectMember/Building	BuildingType
ID		NUMBER	@gml:id	xs:ID
NAME		VARCHAR2		
FUNCTION		VARCHAR2		
YEAR_OF_CONSTRUCTION		NUMBER		
OWNER		VARCHAR2		
MEASURED_HEIGHT		NUMBER		
LOD1_SURFACE_GEOMETRY		SDO_GEOMETRY	lod1MultiSurface	gml:MultiSurfacePropertyType
LOD1_EARTH_SURFACE_GEOMETRY		SDO_GEOMETRY	lod1MultiSurface	gml:MultiSurfacePropertyType
VIEW_TIN		*Table*	cityObjectMember/TINRelief	TINReliefType
THEID		NUMBER	@gml:id	xs:ID
CLASS		VARCHAR2		
GEOM		SDO_GEOMETRY	tin/gml:TriangulatedSurface	gml:TriangulatedSurfaceType

Figure X. Mapping from tables and Views to CityGML using GO Publisher (alt II).

In the second alternative the input tables and views were mapped to reach a similar CityGML output file (see figure X) as in alternative I.

Comparison

It becomes clear when comparing the database input process using the FME workbenches that the geometry and semantics divided table structure (alternative I) is more complex to load data into. Not only that there are a larger number of tables, but also that the constraints forces the geometry to be loaded first. This created a practical problem to be solved within FME. In addition, for each source data file two destination files are mostly needed compared to only one in alternative I. Concerning retrieval of data the testing of mapping to CityGML classes showed that both alternatives requires views for collecting the proper data for a semantic features class. Alternative

I is slightly more complex to retrieve data from while alternative II requires a collection of earth surface features from many tables.

Conclusions

After testing the loading and retrieval of data in both of the database implementations using the chosen dataset none of the alternatives have a strong advantage compared to the other. An advantage of storing all earth surface features in the same table is that retrieval is more easily performed. A collection of all earth surfaces (alternative I) into one table therefore makes sense. To gather all single polygons into the same table (alternative I) to avoid redundant geometry types is however not an advantage since the features are often retrieved according to the semantic features class. Therefore a database implementation using the earth surface storage of alternative I and the semantic-geometric storage of alternative II would be an appropriate solution to avoid the UNION view in alternative II but still keep the simplicity of semantically separated tables including feature geometry.

Appendix A

SQL-scripts for creating tables and metadata for implementation alternative I

```
DELETE FROM USER_SDO_GEOM_METADATA;

DROP TABLE BELOW_SURFACE_SPACE CASCADE CONSTRAINTS
;
DROP TABLE BUILDING CASCADE CONSTRAINTS
;
DROP TABLE CITYFURNITURE CASCADE CONSTRAINTS
;
DROP TABLE CITYOBJECT CASCADE CONSTRAINTS
;
DROP TABLE COMPUND_GEOMETRY CASCADE CONSTRAINTS
;
DROP TABLE CONSTRUCTION_WORK CASCADE CONSTRAINTS
;
DROP TABLE CURVE_GEOMETRY CASCADE CONSTRAINTS
;
DROP TABLE EARTH_SURFACE_GEOMETRY CASCADE CONSTRAINTS
;
DROP TABLE GEOLOGY CASCADE CONSTRAINTS
;
DROP TABLE LANDCOVER CASCADE CONSTRAINTS
;
DROP TABLE POINT_GEOMETRY CASCADE CONSTRAINTS
;
DROP TABLE SURFACE_GEOMETRY CASCADE CONSTRAINTS
;
DROP TABLE TEXTURE_IMAGE CASCADE CONSTRAINTS
;
DROP TABLE TRANSPORTATION CASCADE CONSTRAINTS
;
DROP TABLE UTILITY CASCADE CONSTRAINTS
;
DROP TABLE VEGETATION CASCADE CONSTRAINTS
;
DROP TABLE WATER CASCADE CONSTRAINTS
;

CREATE TABLE BELOW_SURFACE_SPACE (
      ID NUMBER NOT NULL,
      USAGE_CLASS VARCHAR(50),
      SPACE_CLASS VARCHAR(50),
      LOD1_TIS_E_S_GEOMETRY_ID NUMBER,
      LOD2_TIS_E_S_GEOMETRY_ID NUMBER,
      LOD3_TIS_E_S_GEOMETRY_ID NUMBER,
      LOD1_SURFACE_GEOMETRY_ID NUMBER NOT NULL,
      LOD2_SURFACE_GEOMETRY_ID NUMBER,
      LOD3_SURFACE_GEOMETRY_ID NUMBER
)
;

CREATE TABLE BUILDING (
```

```
	ID NUMBER NOT NULL,
	NAME VARCHAR(256),
	FUNCTION VARCHAR(256),
	YEAR_OF_CONSTRUCTION NUMBER,
	OWNER VARCHAR(255),
	ROOF_TYPE NUMBER,
	MEASURED_HEIGHT NUMBER,
	NO_OF_STOREYS_ABOVE_GROUND NUMBER,
	NO_OF_STOREYS_BELOW_GROUND NUMBER,
	LOD1_SURFACE_GEOMETRY_ID NUMBER,
	LOD2_SURFACE_GEOMETRY_ID NUMBER,
	LOD3_SURFACE_GEOMETRY_ID NUMBER,
	LOD1_TIS_E_S_GEOMETRY_ID NUMBER,
	LOD2_TIS_E_S_GEOMETRY_ID NUMBER,
	LOD3_TIS_E_S_GEOMETRY_ID NUMBER
)
;

CREATE TABLE CITYFURNITURE (
	ID NUMBER NOT NULL,
	CITYFURNITURE_CLASS VARCHAR(50),
	HEIGHT NUMBER,
	TIP_POINT_GEOMETRY_ID NUMBER,
	TIC_CURVE_GEOMETRY_ID NUMBER
)
;

CREATE TABLE CITYOBJECT (
	ID NUMBER NOT NULL,
	CLASS_ID NUMBER,
	ENVELOPE MDSYS.SDO_GEOMETRY
)
;

CREATE TABLE COMPUND_GEOMETRY (
	ID NUMBER NOT NULL,
	SURFACE_GEOMETRY_ID NUMBER NOT NULL,
	EARTH_SURFACE_GEOMETRY_ID NUMBER NOT NULL,
	IS_SOLID NUMBER(1) NOT NULL,
	SOLID_GEOMETRY MDSYS.SDO_GEOMETRY
)
;

CREATE TABLE CONSTRUCTION_WORK (
	ID NUMBER NOT NULL,
	CONSTRUCTION_USAGE_CLASS VARCHAR(50),
	CONSTRUCTION_FUNCTION VARCHAR(50),
	LOD1_TIS_E_S_GEOMETRY_ID NUMBER NOT NULL,
	LOD2_TIS_E_S_GEOMETRY_ID NUMBER,
	LOD3_TIS_E_S_GEOMETRY_ID NUMBER,
	LOD1_SURFACE_GEOMETRY_ID NUMBER NOT NULL,
	LOD2_SURFACE_GEOMETRY_ID NUMBER,
	LOD3_SURFACE_GEOMETRY_ID NUMBER
)
;

CREATE TABLE CURVE_GEOMETRY (
```

```
	ID NUMBER NOT NULL,
	SURFACE_GEOMETRY_ID NUMBER NOT NULL,
	TRANSFORMATIONMATRIX VARCHAR(255),
	CURVE_GEOMETRY MDSYS.SDO_GEOMETRY NOT NULL
)
;

CREATE TABLE EARTH_SURFACE_GEOMETRY (
	ID NUMBER NOT NULL,
	SURFACE_ID NUMBER NOT NULL,
	LOD NUMBER(1) NOT NULL,
	THEMATIC_LANDUSE VARCHAR(50),
	IS_CLOSURE_SURFACE NUMBER(1),
	MULTIPOLYGON_GEOMETRY MDSYS.SDO_GEOMETRY NOT NULL,
	TEXTURE_ID NUMBER,
	TEXTURE_DRAPING_PARAMETERS VARCHAR(255),
	IS_REPEATED_TEXTURE NUMBER(1)
)
;

INSERT INTO user_sdo_geom_metadata VALUES('EARTH_SURFACE_GEOMETRY',
'MULTIPOLYGON_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);

CREATE TABLE GEOLOGY (
	ID NUMBER NOT NULL,
	GEOLOGY_CLASS VARCHAR(50),
	TIS_EARTH_SURFACE_GEOMETRY_ID NUMBER NOT NULL,
	SURFACE_GEOMETRY_ID NUMBER,
	TIP_POINT_GEOMETRY_ID NUMBER,
	DEPTH NUMBER
)
;

CREATE TABLE LANDCOVER (
	ID NUMBER NOT NULL,
	LANDCOVER_GROUND_CLASS VARCHAR(50),
	USAGE_CLASS VARCHAR(50),
	EARTH_SURFACE_GEOMETRY_ID NUMBER NOT NULL
)
;

CREATE TABLE POINT_GEOMETRY (
	ID NUMBER NOT NULL,
	SURFACE_GEOMETRY_ID NUMBER NOT NULL,
	TRANSFORMATIONMATRIX VARCHAR(255),
	POINT_GEOMETRY MDSYS.SDO_GEOMETRY NOT NULL
)
;

CREATE TABLE SURFACE_GEOMETRY (
	ID NUMBER NOT NULL,
	SURFACE_ID NUMBER NOT NULL,
	FRONT_TEXTURE_ID NUMBER,
	FRONT_TEXTURE_COORIDINATES MDSYS.SDO_GEOMETRY,
```

```
      FRONT_COLOR_RED NUMBER,
      FRONT_COLOR_GREEN NUMBER,
      FRONT_COLOR_BLUE NUMBER,
      BACK_TEXTURE_ID NUMBER,
      BACK_TEXTURE_COORDINATES MDSYS.SDO_GEOMETRY,
      BACK_COLOR_RED NUMBER,
      BACK_COLOR_GREEN NUMBER,
      BACK_COLOR_BLUE NUMBER,
      FRONT_OPACITY NUMBER,
      BACK_OPACITY NUMBER,
      IS_CLOSURESURFACE NUMBER(1) NOT NULL,
      POLYGON_GEOMETRY MDSYS.SDO_GEOMETRY NOT NULL,
      IS_REPEATED_TEXTURE NUMBER(1)
)
;

INSERT INTO user_sdo_geom_metadata VALUES('SURFACE_GEOMETRY',
'POLYGON_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);

CREATE TABLE TEXTURE_IMAGE (
      ID NUMBER NOT NULL,
      TEXTURE_IMAGE ORDSYS.ORDIMAGE
)
;

CREATE TABLE TRANSPORTATION (
      ID NUMBER NOT NULL,
      TRANSPORTATION_CLASS VARCHAR(50),
      TRANSPORTATION_S_MATERIAL VARCHAR(50),
      EARTH_SURFACE_GEOMETRY_ID NUMBER NOT NULL
)
;

CREATE TABLE UTILITY (
      ID NUMBER NOT NULL,
      UTILITY_CLASS VARCHAR(50),
      DIAMETER NUMBER,
      POINT_GEOMETRY_ID NUMBER,
      CURVE_GEOMETRY_ID NUMBER,
      SURFACE_GEOMETRY_ID NUMBER,
      IS_ABOVE_SURFACE NUMBER(1) NOT NULL,
      HEIGHT NUMBER,
      WIDTH NUMBER
)
;

CREATE TABLE VEGETATION (
      ID NUMBER NOT NULL,
      PLANT_CLASS VARCHAR(50),
      HEIGHT NUMBER,
      PLANT_DISTANCE NUMBER,
      IS_SOLID NUMBER(1) NOT NULL,
      TIP_POINT_GEOMETRY_ID NUMBER,
      TIC_CURVE_GEOMETRY_ID NUMBER,
```

```
      TIS_EARTH_SURFACE_GEOMETRY_ID NUMBER,
      SURFACE_GEOMETRY_ID NUMBER
)
;

CREATE TABLE WATER (
      ID NUMBER NOT NULL,
      WATER_CLASS VARCHAR(50),
      WATER_FUNCTION VARCHAR(50),
      TIS_EARTH_SURFACE_GEOMETRY_ID NUMBER NOT NULL,
      SURFACE_GEOMETRY_ID NUMBER
)
;


ALTER TABLE BELOW_SURFACE_SPACE ADD CONSTRAINT PK_UNDERGROUND_SPACE
      PRIMARY KEY (ID)
;

ALTER TABLE BUILDING ADD CONSTRAINT PK_BUILDING
      PRIMARY KEY (ID)
;

ALTER TABLE CITYFURNITURE ADD CONSTRAINT PK_POINT_CITYFURNITURE
      PRIMARY KEY (ID)
;

ALTER TABLE CITYOBJECT ADD CONSTRAINT PK_CITYOBJECT
      PRIMARY KEY (ID)
;

ALTER TABLE COMPUND_GEOMETRY ADD CONSTRAINT PK_3D_BODY
      PRIMARY KEY (ID)
;

ALTER TABLE CONSTRUCTION_WORK ADD CONSTRAINT PK_CONSTRUCTION_WORK
      PRIMARY KEY (ID)
;

ALTER TABLE CURVE_GEOMETRY ADD CONSTRAINT PK_CURVE_GEOMETRY
      PRIMARY KEY (ID)
;

ALTER TABLE EARTH_SURFACE_GEOMETRY ADD CONSTRAINT
PK_EARTH_SURFACE_OBJECT
      PRIMARY KEY (ID, SURFACE_ID)
;

ALTER TABLE GEOLOGY ADD CONSTRAINT PK_GEOLOGIC_FEATURE
      PRIMARY KEY (ID)
;

ALTER TABLE LANDCOVER ADD CONSTRAINT PK_LANDCOVER
      PRIMARY KEY (ID)
;

ALTER TABLE POINT_GEOMETRY ADD CONSTRAINT PK_3D_POINT
```

```
      PRIMARY KEY (ID)
;

ALTER TABLE SURFACE_GEOMETRY ADD CONSTRAINT PK_SURFACE_GEOMETRY
      PRIMARY KEY (ID, SURFACE_ID)
;

ALTER TABLE TEXTURE_IMAGE ADD CONSTRAINT PK_TEXTURE_IMAGE
      PRIMARY KEY (ID)
;

ALTER TABLE TRANSPORTATION ADD CONSTRAINT PK_TRANSPORTATION
      PRIMARY KEY (ID)
;

ALTER TABLE UTILITY ADD CONSTRAINT PK_POINT_UTILITY
      PRIMARY KEY (ID)
;

ALTER TABLE VEGETATION ADD CONSTRAINT PK_POINT_PLANT
      PRIMARY KEY (ID)
;

ALTER TABLE WATER ADD CONSTRAINT PK_WATERBODY
      PRIMARY KEY (ID)
;



ALTER TABLE CITYFURNITURE ADD CONSTRAINT FK_TIC_CURVE_GEOMETRY_ID_ID
      FOREIGN KEY (TIC_CURVE_GEOMETRY_ID) REFERENCES CURVE_GEOMETRY
(ID)
;

ALTER TABLE CITYFURNITURE ADD CONSTRAINT FK_TIP_POINT_GEOMETRY_ID_ID
      FOREIGN KEY (TIP_POINT_GEOMETRY_ID) REFERENCES POINT_GEOMETRY
(ID)
;

ALTER TABLE EARTH_SURFACE_GEOMETRY ADD CONSTRAINT FK_TEXTURE_ID_ID
      FOREIGN KEY (TEXTURE_ID) REFERENCES TEXTURE_IMAGE (ID)
;

ALTER TABLE GEOLOGY ADD CONSTRAINT FK_POINT_GEOMETRY_ID_ID
      FOREIGN KEY (TIP_POINT_GEOMETRY_ID) REFERENCES POINT_GEOMETRY
(ID)
;

ALTER TABLE SURFACE_GEOMETRY ADD CONSTRAINT FK_BACK_TEXTURE_ID_ID
      FOREIGN KEY (BACK_TEXTURE_ID) REFERENCES TEXTURE_IMAGE (ID)
;

ALTER TABLE SURFACE_GEOMETRY ADD CONSTRAINT FK_FRONT_TEXTURE_ID_ID
      FOREIGN KEY (FRONT_TEXTURE_ID) REFERENCES TEXTURE_IMAGE (ID)
;

ALTER TABLE UTILITY ADD CONSTRAINT FK_POINT_GEOMETRY_ID_ID_2
```

```
      FOREIGN KEY (POINT_GEOMETRY_ID) REFERENCES POINT_GEOMETRY (ID)
;

ALTER TABLE UTILITY ADD CONSTRAINT FK_CURVE_GEOMETRY_ID_ID
      FOREIGN KEY (CURVE_GEOMETRY_ID) REFERENCES CURVE_GEOMETRY (ID)
;

ALTER TABLE VEGETATION ADD CONSTRAINT FK_TIC_CURVE_GEOMETRY_ID_ID_2
      FOREIGN KEY (TIC_CURVE_GEOMETRY_ID) REFERENCES CURVE_GEOMETRY
(ID)
;

ALTER TABLE VEGETATION ADD CONSTRAINT FK_TIS_POINT_GEOMETRY_ID_ID
      FOREIGN KEY (TIP_POINT_GEOMETRY_ID) REFERENCES POINT_GEOMETRY
(ID)
;
```

SQL-scripts for creating tables and metadata for implementation alternative II

```
create or replace TYPE TEXTURE AS OBJECT (
  ID NUMBER,
  IMAGE BLOB
)

create or replace TYPE SYMBOL AS OBJECT (
  ID NUMBER,
  GEOMETRY BLOB
)

DELETE FROM user_sdo_geom_metadata;

DROP TABLE BELOW_SURFACE_SPACE CASCADE CONSTRAINTS
;
DROP TABLE BELOW_SURFACE_SURFACE CASCADE CONSTRAINTS
;
DROP TABLE BUILDING CASCADE CONSTRAINTS
;
DROP TABLE BUILDING_SURFACE CASCADE CONSTRAINTS
;
DROP TABLE CITYFURNITURE CASCADE CONSTRAINTS
;
DROP TABLE CITYOBJECT CASCADE CONSTRAINTS
;
DROP TABLE CONSTRUCTION_WORK CASCADE CONSTRAINTS
;
DROP TABLE CONSTRUCTION_WORK_SURFACE CASCADE CONSTRAINTS
;
DROP TABLE GEOLOGY CASCADE CONSTRAINTS
;
DROP TABLE LANDCOVER CASCADE CONSTRAINTS
```

```
;
DROP TABLE TRANSPORTATION CASCADE CONSTRAINTS
;
DROP TABLE UTILITY CASCADE CONSTRAINTS
;
DROP TABLE VEGETATION CASCADE CONSTRAINTS
;
DROP TABLE WATER CASCADE CONSTRAINTS
;

CREATE TABLE BELOW_SURFACE_SPACE (
      ID NUMBER NOT NULL,
      USAGE_CLASS VARCHAR(50),
      SPACE_CLASS VARCHAR(50),
      EARTH_SURFACE_GEOMETRY MDSYS.SDO_GEOMETRY,
      SURFACE_GEOMETRY_ID NUMBER,
      REPEATED_TEXTURE_IMAGE TEXTURE,
      SOLID_GEOMETRY MDSYS.SDO_GEOMETRY
)
;

INSERT INTO user_sdo_geom_metadata VALUES('BELOW_SURFACE_SPACE',
'EARTH_SURFACE_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);

CREATE TABLE BELOW_SURFACE_SURFACE (
      ID NUMBER NOT NULL,
      SURFACE_ID NUMBER,
      FRONT_TEXTURE TEXTURE,
      FRONT_TEXTURE_COORDINATES MDSYS.SDO_GEOMETRY,
      FRONT_COLOR_RED NUMBER,
      FRONT_COLOR_GREEN NUMBER,
      FRONT_COOR_BLUE NUMBER,
      BACK_TEXTURE_COORDINATES MDSYS.SDO_GEOMETRY,
      BACK_COLOR_RED NUMBER,
      BACK_COLOR_GREEN NUMBER,
      BACK_COLOR_BLUE NUMBER,
      FRONT_OPACITY NUMBER,
      BACK_OPACITY NUMBER,
      IS_CLOSURESURFACE NUMBER(1),
      POLYGON_GEOMETRY MDSYS.SDO_GEOMETRY
)
;

INSERT INTO user_sdo_geom_metadata VALUES('BELOW_SURFACE_SURFACE',
'POLYGON_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);

CREATE TABLE BUILDING (
      ID NUMBER NOT NULL,
      NAME VARCHAR(256),
      FUNCTION VARCHAR(256),
      YEAR_OF_CONSTRUCTION NUMBER,
```

```
	OWNER VARCHAR(255),
	ROOF_TYPE NUMBER,
	MEASURED_HEIGHT NUMBER,
	NO_OF_STOREYS_ABOVE_GROUND NUMBER,
	NO_OF_STOREYS_BELOW_GROUND NUMBER,
	LOD1_SURFACE_GEOMETRY_ID NUMBER,
	LOD2_SURFACE_GEOMETRY_ID NUMBER,
	LOD3_SURFACE_GEOMETRY_ID NUMBER,
	LOD1_EARTH_SURFACE_GEOMETRY MDSYS.SDO_GEOMETRY,
	LOD2_EARTH_SURFACE_GEOMETRY MDSYS.SDO_GEOMETRY,
	LOD3_EARTH_SURFACE_GEOMETRY MDSYS.SDO_GEOMETRY,
	SOLID_GEOMETRY MDSYS.SDO_GEOMETRY
)
;

INSERT INTO user_sdo_geom_metadata VALUES('BUILDING',
'LOD1_EARTH_SURFACE_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);


CREATE TABLE BUILDING_SURFACE (
	ID NUMBER NOT NULL,
	SURFACE_ID NUMBER NOT NULL,
	FRONT_TEXTURE TEXTURE,
	FRONT_TEXTURE_COORDINATES MDSYS.SDO_GEOMETRY,
	FRONT_COLOR_RED NUMBER,
	FRONT_COLOR_GREEN NUMBER,
	FRONT_COLOR_BLUE NUMBER,
	BACK_TEXTURE TEXTURE,
	BACK_TEXTURE_COORDINATES MDSYS.SDO_GEOMETRY,
	BACK_COLOR_RED NUMBER,
	BACK_COLOR_GREEN NUMBER,
	BACK_COLOR_BLUE NUMBER,
	FRONT_OPACITY NUMBER,
	BACK_OPACITY NUMBER,
	IS_CLOSURESURFACE NUMBER(1) NOT NULL,
	POLYGON_GEOMETRY MDSYS.SDO_GEOMETRY NOT NULL
)
;

INSERT INTO user_sdo_geom_metadata VALUES('BUILDING_SURFACE',
'POLYGON_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);

CREATE TABLE CITYFURNITURE (
	ID NUMBER NOT NULL,
	CITYFURNITURE_CLASS VARCHAR(50),
	HEIGHT NUMBER,
	LOD0_GEOMETRY MDSYS.SDO_GEOMETRY,
	LOD1_4_GEOMETRY SYMBOL,
	REPREATED_TEXTURE TEXTURE
)
;
```

```
INSERT INTO user_sdo_geom_metadata VALUES('CITYFURNITURE',
'LOD0_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);

CREATE TABLE CITYOBJECT (
      ID NUMBER NOT NULL,
      CLASS_ID NUMBER,
      ENVELOPE MDSYS.SDO_GEOMETRY
)
;

CREATE TABLE CONSTRUCTION_WORK (
      ID NUMBER NOT NULL,
      CONSTRUCTION_USAGE_CLASS VARCHAR(50),
      CONSTRUCTION_FUNCTION VARCHAR(50),
      LOD0_SURFACE_GEOMETRY_ID NUMBER,
      LOD1_SURFACE_GEOMETRY_ID NUMBER,
      LOD2_SURFACE_GEOMETRY_ID NUMBER,
      LOD0_EARTH_SURFACE_GEOMETRY MDSYS.SDO_GEOMETRY,
      LOD1_EARTH_SURFACE_GEOMETRY MDSYS.SDO_GEOMETRY,
      LOD2_EARTH_SURFACE_GEOMETRY MDSYS.SDO_GEOMETRY
)
;

INSERT INTO user_sdo_geom_metadata VALUES('CONSTRUCTION_WORK',
'LOD0_EARTH_SURFACE_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);

CREATE TABLE CONSTRUCTION_WORK_SURFACE (
      ID NUMBER NOT NULL,
      SURFACE_ID NUMBER NOT NULL,
      FRONT_TEXTURE TEXTURE,
      FRONT_TEXTURE_COORDINATES MDSYS.SDO_GEOMETRY,
      FRONT_COLOR_RED NUMBER,
      FRONT_COLOR_GREEN NUMBER,
      FRONT_COLOR_BLUE NUMBER,
      BACK_TEXTURE TEXTURE,
      BACK_TEXTURE_COORDINATES MDSYS.SDO_GEOMETRY,
      BACK_COLOR_RED NUMBER,
      BACK_COLOR_GREEN NUMBER,
      BACK_COLOR_BLUE NUMBER,
      FRONT_OPACITY NUMBER,
      BACK_OPACITY NUMBER,
      IS_CLOSURESURFACE NUMBER NOT NULL,
      POLYGON_GEOMETRY MDSYS.SDO_GEOMETRY NOT NULL
)
;

INSERT INTO user_sdo_geom_metadata VALUES('CONSTRUCTION_WORK_SURFACE',
'POLYGON_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
```

```
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);

CREATE TABLE GEOLOGY (
      ID NUMBER NOT NULL,
      GEOLOGY_CLASS VARCHAR(50),
      EARTH_SURFACE_GEOMETRY MDSYS.SDO_GEOMETRY,
      LOD1_4_GEOMETRY MDSYS.SDO_GEOMETRY,
      LOD0_GEOMETRY MDSYS.SDO_GEOMETRY,
      DEPTH NUMBER,
      SOLID_GEOMETRY MDSYS.SDO_GEOMETRY,
      REPEATED_TEXTURE TEXTURE
)
;

INSERT INTO user_sdo_geom_metadata VALUES('GEOLOGY', 'LOD1_4_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);

CREATE TABLE LANDCOVER (
      ID NUMBER NOT NULL,
      LANDCOVER_GROUND_CLASS VARCHAR(50),
      USAGE_CLASS VARCHAR(50),
      EARTH_SURFACE_GEOMETRY MDSYS.SDO_GEOMETRY NOT NULL,
      THEMATIC_LANDUSE VARCHAR(50),
      REPEATED_TEXTURE TEXTURE
)
;

INSERT INTO user_sdo_geom_metadata VALUES('LANDCOVER',
'EARTH_SURFACE_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);

CREATE TABLE TRANSPORTATION (
      ID NUMBER NOT NULL,
      TRANSPORTATION_CLASS VARCHAR(50),
      TRANSPORTATION_S_MATERIAL VARCHAR(50),
      EARTH_SURFACE_GEOMETRY MDSYS.SDO_GEOMETRY NOT NULL,
      THEMATIC_LANDUSE VARCHAR(50),
      REPEATED_TEXTURE TEXTURE
)
;

INSERT INTO user_sdo_geom_metadata VALUES('TRANSPORTATION',
'EARTH_SURFACE_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);

CREATE TABLE UTILITY (
      ID NUMBER NOT NULL,
      UTILITY_CLASS VARCHAR(50),
      DIAMETER NUMBER,
      TRANSFORMATION_MATRIX VARCHAR(255),
      LOD0_GEOMETRY MDSYS.SDO_GEOMETRY,
```

```
      LOD1_4_SURFACE_GEOMETRY MDSYS.SDO_GEOMETRY,
      LOD1_4_SYMBOL_GEOMETRY SYMBOL,
      IS_ABOVE_SURFACE NUMBER(1) NOT NULL,
      REPEATED_TEXTURE TEXTURE
)
;

INSERT INTO user_sdo_geom_metadata VALUES('UTILITIY', 'LOD0_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);

CREATE TABLE VEGETATION (
      ID NUMBER NOT NULL,
      PLANT_CLASS VARCHAR(50),
      HEIGHT NUMBER,
      PLANT_DISTANCE NUMBER,
      IS_SOLID NUMBER(1) NOT NULL,
      TRANSFORMATION_MATRIX VARCHAR(255),
      LOD0_GEOMETRY MDSYS.SDO_GEOMETRY,
      LOD1_GEOMETRY MDSYS.SDO_GEOMETRY,
      LOD2_4_GEOMETRY SYMBOL,
      REPEATED_TEXTURE TEXTURE,
      TEXTURE_COORDINATES MDSYS.SDO_GEOMETRY
)
;

INSERT INTO user_sdo_geom_metadata VALUES('VEGETATION',
'LOD0_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);

CREATE TABLE WATER (
      ID NUMBER NOT NULL,
      WATER_CLASS VARCHAR(50),
      WATER_FUNCTION VARCHAR(50),
      GROUND_SURFACE_GEOMETRY MDSYS.SDO_GEOMETRY,
      EARTH_SURFACE_GEOMETRY MDSYS.SDO_GEOMETRY,
      CLOSURE_SURFACE_GEOMETRY MDSYS.SDO_GEOMETRY,
      SOLID_GEOMETRY MDSYS.SDO_GEOMETRY
)
;

INSERT INTO user_sdo_geom_metadata VALUES('WATER',
'GROUND_SURFACE_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);

INSERT INTO user_sdo_geom_metadata VALUES('WATER',
'EARTH_SURFACE_GEOMETRY',
sdo_dim_array( sdo_dim_element('X', 80000,90000, 0.005),
sdo_dim_element('Y', 446000,447000, 0.005),
sdo_dim_element('Z', -1000,1000, 0.005)), NULL);
```

```
ALTER TABLE BELOW_SURFACE_SPACE ADD CONSTRAINT PK_UNDERGROUND_SPACE
      PRIMARY KEY (ID)
;

ALTER TABLE BELOW_SURFACE_SURFACE ADD CONSTRAINT
PK_BELOW_SURFACE_SURFACE
      PRIMARY KEY (ID)
;

ALTER TABLE BUILDING ADD CONSTRAINT PK_BUILDING
      PRIMARY KEY (ID)
;

ALTER TABLE BUILDING_SURFACE ADD CONSTRAINT PK_BUILDING_SURFACE
      PRIMARY KEY (ID)
;

ALTER TABLE CITYFURNITURE ADD CONSTRAINT PK_POINT_CITYFURNITURE
      PRIMARY KEY (ID)
;

ALTER TABLE CITYOBJECT ADD CONSTRAINT PK_CITYOBJECT
      PRIMARY KEY (ID)
;

ALTER TABLE CONSTRUCTION_WORK ADD CONSTRAINT PK_CONSTRUCTION_WORK
      PRIMARY KEY (ID)
;

ALTER TABLE CONSTRUCTION_WORK_SURFACE ADD CONSTRAINT
PK_CONSTRUCTION_WORK_SURFACE
      PRIMARY KEY (ID)
;

ALTER TABLE GEOLOGY ADD CONSTRAINT PK_GEOLOGIC_FEATURE
      PRIMARY KEY (ID)
;

ALTER TABLE LANDCOVER ADD CONSTRAINT PK_LANDCOVER
      PRIMARY KEY (ID)
;

ALTER TABLE TRANSPORTATION ADD CONSTRAINT PK_TRANSPORTATION
      PRIMARY KEY (ID)
;

ALTER TABLE UTILITY ADD CONSTRAINT PK_UTILITIY
      PRIMARY KEY (ID)
;

ALTER TABLE VEGETATION ADD CONSTRAINT PK_POINT_PLANT
      PRIMARY KEY (ID)
;

ALTER TABLE WATER ADD CONSTRAINT PK_WATERBODY
      PRIMARY KEY (ID)
;
```

Appendix B

Views created in implementation alternative I

```
DROP VIEW VIEW_WATER;
DROP VIEW VIEW_WATER_AGGREGATED_SURFACE;
DROP VIEW VIEW_POINT_CITY_FURNITURE;
DROP VIEW VIEW_POINT_VEGETATION;
DROP VIEW VIEW_TRANSPORTATION;
DROP VIEW VIEW_LANDCOVER;
DROP VIEW VIEW_BUILDING_LOD1;
DROP VIEW VIEW_BUILDING_AGGRE_SURFACE;


create or replace view VIEW_BUILDING_LOD1 as (
SELECT a.ID, a.NAME, b.GEOM AS LOD_1_SURFACE_GEOMETRY,
c.MULTIPOLYGON_GEOMETRY AS LOD_1_EARTH_SURFACE_GEOMETRY
FROM BUILDING a, VIEW_BUILDING_AGGRE_SURFACE b, EARTH_SURFACE_GEOMETRY
c
WHERE a.LOD1_TIS_E_S_GEOMETRY_ID = c.SURFACE_ID AND
a.LOD1_SURFACE_GEOMETRY_ID = b.SURFACE_ID
);

create or replace view VIEW_BUILDING_AGGRE_SURFACE as (
SELECT SURFACE_ID as SURFACE_ID, SDO_AGGR_UNION(
SDOAGGRTYPE(d.POLYGON_GEOMETRY, 0.005)) AS GEOM
FROM SURFACE_GEOMETRY d
WHERE SURFACE_ID IN (SELECT LOD1_SURFACE_GEOMETRY_ID FROM BUILDING)
GROUP BY SURFACE_ID
);

create or replace view VIEW_WATER as (
SELECT a.ID, a.WATER_CLASS,a.WATER_FUNCTION, b.GEOM AS
WATER_GROUND_SURFACE_GEOMETRY, c.MULTIPOLYGON_GEOMETRY AS
WATER_EARTH_SURFACE_GEOMETRY
FROM WATER a, VIEW_WATER_AGGREGATED_SURFACE b, EARTH_SURFACE_GEOMETRY c
WHERE a.TIS_EARTH_SURFACE_GEOMETRY_ID = c.SURFACE_ID AND
a.SURFACE_GEOMETRY_ID = b.SURFACE_ID
);

create or replace view VIEW_WATER_AGGREGATED_SURFACE as (
SELECT SURFACE_ID as SURFACE_ID, SDO_AGGR_UNION(
SDOAGGRTYPE(d.POLYGON_GEOMETRY, 0.005)) AS GEOM
FROM SURFACE_GEOMETRY d
WHERE SURFACE_ID IN (SELECT SURFACE_GEOMETRY_ID FROM WATER)
GROUP BY SURFACE_ID
);


create or replace view VIEW_POINT_CITY_FURNITURE as (
SELECT C.ID, C.CITYFURNITURE_CLASS, C.TIP_POINT_GEOMETRY_ID,
P.POINT_GEOMETRY
FROM CITYFURNITURE C, POINT_GEOMETRY P
WHERE C.ID = P.ID
);
```

```
create or replace view VIEW_POINT_VEGETATION as (
SELECT V.ID, V.PLANT_CLASS, V.TIP_POINT_GEOMETRY_ID, P.POINT_GEOMETRY
FROM VEGETATION V, POINT_GEOMETRY P
WHERE V.ID = P.ID
);

create or replace view VIEW_LANDCOVER as (
SELECT L.ID, L.LANDCOVER_GROUND_CLASS, L.USAGE_CLASS,
E.MULTIPOLYGON_GEOMETRY
FROM LANDCOVER L, EARTH_SURFACE_GEOMETRY E
WHERE L.EARTH_SURFACE_GEOMETRY_ID = E.SURFACE_ID
);

create or replace view VIEW_TRANSPORTATION as (
SELECT T.ID, T.TRANSPORTATION_CLASS, T.TRANSPORTATION_S_MATERIAL,
E.MULTIPOLYGON_GEOMETRY
FROM TRANSPORTATION T, EARTH_SURFACE_GEOMETRY E
WHERE T.EARTH_SURFACE_GEOMETRY_ID = E.SURFACE_ID
  );
```

Views created in implementation alternative II

```
DROP VIEW view_solitary_vegetation;
DROP VIEW view_curve_vegetation;
DROP VIEW view_point_city_furniture;
DROP VIEW view_curve_city_furniture;
DROP VIEW VIEW_TIN;
DROP VIEW VIEW_BUILDING_AGGRE_SURFACE;
DROP VIEW VIEW_BUILDING_LOD1;


create or replace view view_solitary_vegetation as (
SELECT * FROM VEGETATION c WHERE c.LOD0_GEOMETRY.Get_GType() ='1');

create or replace view view_point_city_furniture as (
SELECT * FROM CITYFURNITURE c WHERE c.LOD0_GEOMETRY.Get_GType() ='1');

create or replace view VIEW_TIN as (
SELECT ID as theID, TRANSPORTATION_CLASS as CLASS,
EARTH_SURFACE_GEOMETRY AS GEOM FROM TRANSPORTATION
UNION
SELECT ID as theID2, LANDCOVER_GROUND_CLASS as CLASS,
EARTH_SURFACE_GEOMETRY AS GEOM FROM LANDCOVER
UNION
SELECT ID as theID2, CONSTRUCTION_USAGE_CLASS as CLASS,
LOD0_EARTH_SURFACE_GEOMETRY AS GEOM FROM CONSTRUCTION_WORK
);

create or replace view VIEW_BUILDING_AGGRE_SURFACE as (
SELECT SURFACE_ID as SURFACE_ID, SDO_AGGR_UNION(
SDOAGGRTYPE(d.POLYGON_GEOMETRY, 0.005)) AS GEOM
FROM BUILDING_SURFACE d
GROUP BY SURFACE_ID
```

```
);

create or replace view VIEW_BUILDING_LOD1 as (
SELECT BUILDING.ID,BUILDING.NAME, BUILDING.FUNCTION,
BUILDING.YEAR_OF_CONSTRUCTION, BUILDING.OWNER, BUILDING.MEASURED_HEIGHT
, VIEW_BUILDING_AGGRE_SURFACE.GEOM AS LOD1_SURFACE_GEOMETRY,
BUILDING.LOD1_EARTH_SURFACE_GEOMETRY
FROM BUILDING, VIEW_BUILDING_AGGRE_SURFACE
WHERE
BUILDING.LOD1_SURFACE_GEOMETRY_ID=VIEW_BUILDING_AGGRE_SURFACE.SURFACE_I
D
);
```