MANAGING 3D GEO SPATIAL DATA
CASE STUDY: SULEYMANIYE REGION OF ISTANBUL HISTORIC PENINSULAR

Alias Abdul Rahman[1], S. Awanis Zulkifli[1], Ismail R. Karas[2], Ibrahim Baz[2] and Gurcan Buyuksalih[3]

[1]Dept of Geoinformatics,
Universiti Teknologi Malaysia,
Johor, Malaysia

[2]Gebze Institute of Technology,
Kocaeli, Turkey

[3]IBB (Bimtas), Altunizade,
Istanbul, Turkey

{alias,awanis}@fksg.utm.my
ragib@gyte.edu.tr
ibaz@gyte.edu.tr
gbuyuksalih@yahoo.com

ABSTRACT

Currently, variety of software is already capable of handling a wide range of spatial problems, beginning with approaches for describing spatial objects to quite complex analysis and 3D visualisation. However, increasing number of applications need more advanced tools for representing and analyzing the 3D world. Among all types of systems dealing with spatial information, GIS has proven to be the most sophisticated system that operates with the largest scope of objects (spatial and semantic), relationships and provide means to analyze them. However, how about 3D GIS? It is the aim of this paper to find the answer by analyzing the available software. An overview of several commercial systems and a 3D case study performed in Oracle Spatial and MicroStation Bentley Map provides knowledge about the 3D functionality offered by commercial systems. At the end, the paper addresses some of the issues and problems involved in developing such a system and recommends directions for further research

1. INTRODUCTION

Nowadays, GIS users are getting more complex datasets and needs to manipulate and generate information as we perceived in the real world, i.e. in 3D environment. This environment provides better understanding of geospatial pattern and phenomena either in small or large areas. Relevant questions such as: How we model the real world objects, i.e. 2D and 3D spatial objects in DBMS and visualizing the objects in GIS/CAD front-end? Answering this question provides an interesting experiment toward realizing 3D geospatial database and information.

In general, many related agencies such as National Mapping Agency (NMA) realize the needs for a central DBMS: a system that spatial and attribute data are maintained in one integrated environment. Many DBMSs are capable of maintaining spatial data in 2D, since spatial data types and data structure in 2D have been implemented in DBMSs. However, aspect of 3D data still needs a lot of efforts and to be addressed. We have seen more and more CAD and GIS software begin to offer interesting functions and solutions towards 3D data management. Integrating or incorporating the two software (i.e. CAD and DBMS) seems able to partly solve the 3D spatial data management issues, still many issues remain difficult to realize at least for the time being.

This paper addresses on how to manage 3D spatial objects using Oracle Spatial and Bentley Map, all experiments are highlighted in this paper. Section 2 elaborates the method on how to represent the 2D and 3D spatial objects in the DBMS. In section 3, the logical data structure of the represented objects in DBMS is discussed. Section 4 focuses on the experiments i.e. the storage of datasets in DBMS and visualization the objects. Finally, we conclude the work with some future tasks that needs to be addressed.

2. MODELLING THE REAL WORLD OBJECTS

The real world objects can be described into two components, i.e. 2D objects such as lot parcels and 3D objects (e.g. buildings). Real world objects are characterized based on certain representation and storage within the DBMS. The representation depends on the purpose of the application of the database, identification and data capture process.

2.1 2D Objects

In general, 2D objects could be represented by a series of polygon geometry, and their nodes as list of points i.e.

connected one-by-one by straight line segments and the last point connected to the first.

For example, to illustrate 2D lot parcels in 2D DBMS are mostly represented by closed polygons, as this will simplify objects identification in GIS. Most lot parcels are identified and capture as a closed polygon as shown at figure 1.

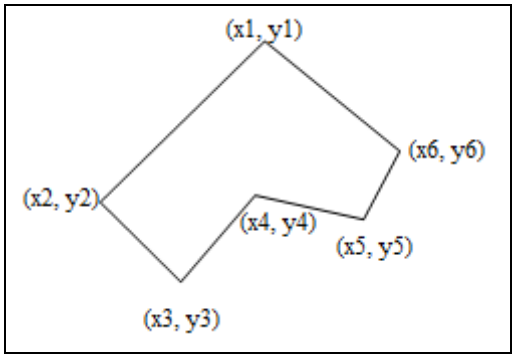


Figure 1: 2D object by closed polygon

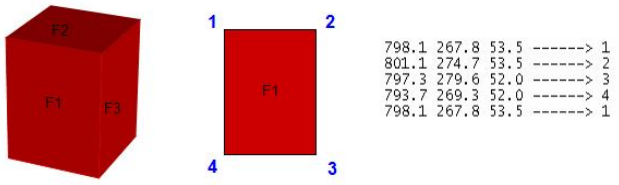
2.2 3D Objects

The representation of real world objects is far more complicated when we move to 3D application. In 3D, opposed to 2.5D, it is not possible to assume that objects can be flattened and defined as polygon footprint on the surface. Instead, a proper 3D model representation of the object is needed, meaning more than one Z-value is attached to a 2D polygon and more information is needed than one Z-value attached to the vertices of the 2D polygon (Zlatanova, 2000).

In 3D, the definition of the geometrical and topological description and validation of the correctness is far more complicated than in 2D. The polyhedral approach, as described in among others (Stoter, 2004) defines the polygons, where each polygon should be 'flat' and valid according to the Simple Feature Specification (SFS) of the Open Geospatial Consortium (OGC), i.e. non-self intersecting. The set of faces should enclose a single volume.

For example, in this research Oracle Spatial 10g are used to manage 3D points, lines and polygons. Using 3D polygons, 3D spatial objects can be represented as polyhedron in two ways: face-by-face polygons or data type multipolygon collection.

In the first method, 3D objects are defined as face-by-face polygons. This model is partly a topological model; since the body is defined by references to the faces (faces share two bodies). To develop 3D objects, series of coordinates that contain x, y, z are extracted from one face of object to another face of object. The 3D object was stored as a list of 3D polygon and it defined by reference to the faces and the faces can be shared by neighbour-bodies as shown at Figure 2. The generated tables for stored 3D object face-by-face look as shown in Table 1.



(a) (b) (c)

Figure 2: The extracted coordinates (face by face) of the 3D object, (a) Build_ID: 1, (b) Face_ID: 1, (c) list of coordinate for face 1.

Build_ID	Face_ID	Shape(mdsys.sdo_geometry)
1	1	x1,y1,z1 x3,y3,z3 x4,y4,z4 x2,y2,z2 x1,y1,z1
	2	x7,y7,z7 x1,y1,z1 x2,y2,z2 x5,y5,z5 x7,y7,z7
	3	x2,y2,z2 x4,y4,z4 x6,y6,z6 x5,y5,z5 x2,y2,z2
	4	x7,y7,z7 x8,y8,z8 x6,y6,z6 z5,y5,z5 x7,y7,z7
	5	x1,y1,z1 x3,y3,z3 x8,y8,z8 z7,y7,z7 x1,y1,z1
	6	x8,y8,z8 x3,y3,z3 x4,y4,z4 z6,y6,z6 x8,y8,z8

Table 1: Coordinates list face by face

In second method defining 3D object in multipolygon collection, a 3D object is stored as one record instead of a set of records. The multipolygon, which is also supported in Oracle Spatial, is used for this representation. This has also been implemented and the example to stored 3D objects using multipolygon method as shown in Figure 3.

```
mdsys.sdo_geometry(3007, NULL, NULL,
mdsys.sdo_elem_info_array(
1, 1003, 1,
16, 1003, 1,
31, 1003, 1,
46, 1003, 1,
61, 1003, 1,
76, 1003, 1
),
SDO_ORDINATE_ARRAY(
x4,y4,z4, ,x3,y3,z3, x2,y2,z2, x1,y1,z1, x4,y4,z4,
x3,y3,z3, ,x4,y4,z4, x8,y8,z8, x7,y7,z7, x3,y3,z3,
x4,y4,z4, ,x1,y1,z1, x5,y5,z5, x8,y8,z8, x4,y4,z4,
x1,y1,z1, ,x2,y2,z2, x6,y6,z6, z5,y5,z5, x1,y1,z1,
x3,y3,z3, ,x2,y2,z2, x6,y6,z6, z7,y7,z7, x3,y3,z3,
x5,y5,z5, ,x6,y6,z6, x7,y7,z7, z8,y8,z8, x5,y5,z5
)));
```

Figure 3: Representation of 3D object using multipolygon method.

An advantage of face by face polygons is that it is recognized as one object by front-end applications (GIS/CAD) that can access, visualize, and edit these data and post the changes back to the database so we can easily store the information about the object i.e. face 1 represent the wall and face 2 represent the roof.

An advantage of 3D multipolygons (compare to face by face polygons) is that it is recognized as one object by front-end applications (GIS/CAD) that can access, visualize, and edit these data and post the changes back to the database. Another advantage of the 3D multipolygon approach is the one-to-one correspondence between a record and an object.

3. MANAGING 3D SPATIAL OBJECTS

3.1 Study Area

Suleymaniye, Istanbul, Turkey is the study area. It is a place of historical objects and monuments. In this area, we assume the following scenario: the user has 3D data organized only in a database (a quite common case for real world data), i.e. no file with graphical information (e.g. DGN) exists. We have experimented with a set of buildings from the Suleymaniye area. Planar rectangular faces constitute each building. The data are build in CAD and further converted to the geometry representation of Oracle Spatial 10g. The conversion is completed with a topology-geometry of the 2D and 3D spatial object. Since the Oracle Spatial 10g geometry does not maintain a true 3D object, we represented every building as a set of faces (walls, flat roofs and foundations). The faces are stored as polygons with 3D coordinates. In this experiment table, namely LOT stored 2D object, table BUILDING represent 3D object stored face by face and table BODY_BUILDING stored the whole 3D object as a one object.

3.2 The physical model

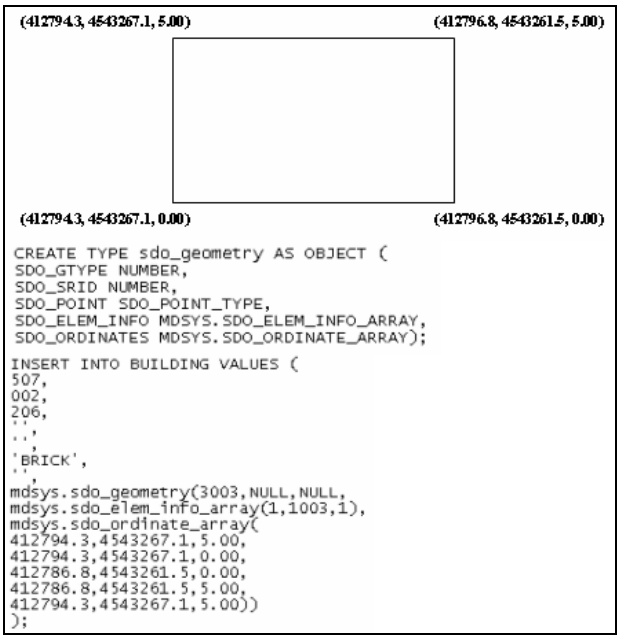
In this experiment two tables that represent 3D objects are develop as shown in Table 2. In the table, 'BODY_BUILDING' the 3D spatial object is defined by a set of records. This table was constructed by using multipolygon method. In the table 'BUILDING', the actual geometries of faces are stored as face-by-face method.

Entity Name	Geometry Type	Field Name	Description
Body_Building	Polygon	Build_Id	Id of the building
		Build_Name	Name of the building
		Age	Age of the building
		No_Lot	Number of the lot
		Build_Shape	Column of Geometry
Building	Polygon	Face_Id	Id of the face
		Material	Material of the wall or face
		Description	Description of the wall or face
		Build_Id	Id of the building
		Face_Shape	Column Geometry
		Build_Area	Area of the building floor
		B_Perimeter	Perimeter of the building

Table 2: Description of BODY_BUILDING and BUILDING table.

Storing of 3D object using face-by-face method

The 3D objects in table 'BUILDING' are stored using face-by-face method and the example to store the 3D object in Oracle as shown at Figure 4.




```
CREATE TYPE sdo_geometry AS OBJECT (
SDO_GTYPE NUMBER,
SDO_SRID NUMBER,
SDO_POINT SDO_POINT_TYPE,
SDO_ELEM_INFO MDSYS.SDO_ELEM_INFO_ARRAY,
SDO_ORDINATES MDSYS.SDO_ORDINATE_ARRAY);

INSERT INTO BUILDING VALUES (
507,
002,
206,
'',
'',
'BRICK',
'',
mdsys.sdo_geometry(3003,NULL,NULL,
mdsys.sdo_elem_info_array(1,1003,1),
mdsys.sdo_ordinate_array(
412794.3,4543267.1,5.00,
412794.3,4543267.1,0.00,
412786.8,4543261.5,0.00,
412786.8,4543261.5,5.00,
412794.3,4543267.1,5.00))
);
```

Figure 4: The 3D Face store by faces

Storing of 3D object using multipolygon method

In the table 'BODY_BUILDING', the 3D objects are stored by multipolygon method and the example to store the 3D objects in Oracle as shown in figure 5.

```
INSERT INTO BODY_BUILDING VALUES (
507,
001,
'KATIP SEMSETINE MOSQUE',
'',
'BRICK',
'',
mdsys.sdo_geometry(3007,NULL,NULL,
mdsys.sdo_elem_info_array(
1,1003,1,
16, 1003, 1,
31, 1003, 1,
46, 1003, 1,
61, 1003, 1,
76, 1003, 1),
mdsys.sdo_ordinate_array(
412802.9,4543265.6,0.00,412794.6,4543269.4,0.00,412798.2,4543278.8,0.00,
412810.9,4543277.7,0.00,412811.6,4543272.3,0.00,412804.9,4543271.9,0.00,
412802.9,4543265.6,0.00,412802.9,4543265.6,11.0,412804.9,4543271.9,11.0,
412811.6,4543272.3,11.0,412810.9,4543277.7,11.0,412798.2,4543278.8,11.0,
412794.6,4543269.4,11.0,412802.9,4543265.6,11.0,412798.1,4543267.8,12.5,
412801.1,4543274.7,12.5,412797.3,4543279.6,11.0,412793.7,4543269.3,11.0,
412798.1,4543267.8,12.5,412801.1,4543274.7,12.5,412811.0,4543274.8,12.5,
412811.1,4543278.6,11.0,412797.3,4543279.6,11.0,412801.1,4543274.7,12.5,
412801.1,4543274.7,12.5,412798.1,4543267.8,12.5,412803.3,4543264.9,11.0,
412805.9,4543270.9,11.0,412801.1,4543274.7,12.5,412811.0,4543274.8,12.5,
412801.1,4543274.7,12.5,412805.5,4543270.9,11.0,412811.9,4543271.9,11.0,
412811.0,4543274.8,12.5,412798.1,4543267.8,12.5,412793.7,4543269.3,11.0,
412803.3,4543264.9,11.0,412798.1,4543267.8,12.5,412811.0,4543274.8,12.5,
412811.9,4543271.9,11.0,412811.1,4543278.6,11.0,412811.0,4543274.8,12.5,
412802.9,4543265.6,0.00,412804.9,4543271.9,0.00,412804.9,4543271.9,11.0,
412802.9,4543265.6,11.0,412802.9,4543265.6,0.00,412804.9,4543271.9,0.00,
412811.6,4543272.3,0.00,412811.6,4543272.3,11.0,412804.9,4543271.9,11.0,
412804.9,4543271.9,0.00,412811.6,4543272.3,0.00,412810.9,4543277.7,0.00,
412810.9,4543277.7,11.0,412811.6,4543272.3,11.0,412811.6,4543272.3,0.00,
412810.9,4543277.7,0.00,412798.2,4543278.8,0.00,412798.2,4543278.8,11.0,
412810.9,4543277.7,11.0,412810.9,4543277.7,0.00,412798.2,4543278.8,0.00,
412794.6,4543269.4,0.00,412794.6,4543269.4,11.0,412798.2,4543278.8,11.0,
412798.2,4543278.8,0.00,412794.6,4543269.4,0.00,412802.9,4543265.6,0.00,
412802.9,4543265.6,11.0,412794.6,4543269.4,11.0,412794.6,4543269.4,0.00,
412803.3,4543264.9,11.0,412805.5,4543270.9,11.0,412811.9,4543271.9,11.0,
412811.1,4543278.6,11.0,412797.3,4543279.6,11.0,412793.7,4543269.3,11.0,
412803.3,4543264.9,11.0)));
```

Figure5: Store 3D object by multipolygon method

3.3 Managing 2D and 3D spatial object in DBMS

In managing 2D and 3D spatial object, Oracle Spatial supports storage for 3D points, lines and polygons. In this experiment, we divide into four steps to store spatial object in DBMS. The first step is creating 2D and 3D tables. In this case, we choose the table 'LOT' to represent 2D objects and the table "BUILDING' to represent 3D objects. The steps to create table for spatial objects as shown in Figure 6.

```
--2D OBJECT--

CREATE TABLE Lot (
NO_LOT NUMBER(5) PRIMARY KEY,
Lot_Area NUMBER(12,3),
Lot_Pmeter NUMBER(12,3),
GEOMETRY MDSYS.SDO_GEOMETRY);

--3D OBJECT--

CREATE TABLE BUILDING (
NO_LOT NUMBER(5),
BUILD_ID NUMBER(5),
FACE_ID NUMBER(4),
BUILD_NAME VARCHAR2(100),
AGE VARCHAR2(10),
MATERIAL VARCHAR2(50),
DESCRIPTION VARCHAR2(50),
GEOMETRY MDSYS.SDO_GEOMETRY);
```

Figure 6: Step 1- creating table in Oracle Spatial

The next steps are inserting the dataset. In Figure 7 represent the SQL- how to insert the 2D and 3D datasets.

```
--2D OBJECT--

INSERT INTO Lot VALUES (
507,
944.820,
130.896,
mdsys.sdo_geometry(2003,NULL,NULL,
mdsys.sdo_elem_info_array(1,1003,1),
mdsys.sdo_ordinate_array(
412797.3000,4543233.6101,412785.9798,4543240.4599,
412786.1302,4543241.0602,412782.7798,4543244.2998,
412781.9001,4543243.7498,412776.5199,4543249.3201,
412775.1699,4543253.9799,412778.7001,4543255.3802,
412781.6599,4543257.2601,412785.8499,4543261.4398,
412793.7298,4543267.9698,412794.6299,4543269.3500,
412795.3899,4543273.0599,412795.6502,4543278.4402,
412810.4601,4543278.8002,412811.3802,4543272.6902,
412813.2699,4543261.6098,412811.2601,4543257.6498,
412810.9299,4543257.0500,412803.1799,4543243.5002,
412800.4600,4543238.9200,412797.3000,4543233.6101))
);

--3D OBJECT--

INSERT INTO BUILDING VALUES (
507,
002,
206,
''
,,'
,
'BRICK',
''
,
mdsys.sdo_geometry(3003,NULL,NULL,
mdsys.sdo_elem_info_array(1,1003,1),
mdsys.sdo_ordinate_array(
412794.3,4543267.1,5.00,
412794.3,4543267.1,0.00,
412786.8,4543261.5,0.00,
412786.8,4543261.5,5.00,
412794.3,4543267.1,5.00))
);
```

Figure7: Step 2 - Insert 2D and 3D data.

The third step is inserting the metadata. Besides the tables that represent the geometries of the objects, metadata is maintained in Oracle by describing the dimension, lower and upper bounds and tolerance in each dimension. In the following figure (Figure 8), the information on the tables (LOT and BUILDING) is inserted in the metadata table. The final step, a spatial index (r-tree in 2D and in 3D) is created in the tables (to speed up spatial queries) as shown in Figure 9.

```
--2D OBJECT--

INSERT INTO USER_SDO_GEOM_METADATA
(TABLE_NAME,
COLUMN_NAME,
DIMINFO,
SRID)
VALUES (
'LOT',
'GEOMETRY',
MDSYS.SDO_DIM_ARRAY(
MDSYS.SDO_DIM_ELEMENT('X', 410000, 420000, 0.005),
MDSYS.SDO_DIM_ELEMENT('Y', 410000, 4600000, 0.005)
),
NULL
);


--3D OBJECT--

INSERT INTO USER_SDO_GEOM_METADATA
(TABLE_NAME,
COLUMN_NAME,
DIMINFO,
SRID)
VALUES (
'BUILDING',
'GEOMETRY',
MDSYS.SDO_DIM_ARRAY(
MDSYS.SDO_DIM_ELEMENT('X', 410000, 420000, 0.005),
MDSYS.SDO_DIM_ELEMENT('Y', 410000, 4600000, 0.005),
MDSYS.SDO_DIM_ELEMENT('Z', 0, 60, 0.005)
),
NULL
);
```

Figure 8: Step 3 - Insert the metadata

```
--2D OBJECT--

CREATE INDEX LOT_IDX
ON LOT(GEOMETRY)
INDEXTYPE IS MDSYS.SPATIAL_INDEX;

--3D OBJECT--

CREATE INDEX BUILDING_IDX
ON BUILDING(GEOMETRY)
INDEXTYPE IS MDSYS.SPATIAL_INDEX
PARAMETERS (' SDO_INDX_DIMS=3 LAYER_GTYPE="POLYGON"');
```

Figure 9: Step 4 - Creating Index.

3.4 The Linkage between Oracle Spatial and Bentley Map

Bentley Map is directly integrated with MicroStation and takes advantage of its features i.e. capture, editing, display, and output capabilities. Bentley support geospatial modelling and interoperability. This software directly reference ESRI SHP files, MapInfo TAB files, Oracle Spatial features and others, or export dataset into these formats for sharing with other projects.

In Bentley Map, topological relationships are stored in the DGN file according to the same model employed by Oracle Spatial. This allows complex analysis operations to occur quickly and maintains compatibility with Oracle Spatial for organizations that choose to use Bentley Map as an editing application for Oracle Spatial.

The object that we store in Oracle will be appearing when we connect the Bentley Map with Oracle Spatial. Figure 10 shows the 2D objects and 3D objects were upload into the Bentley Map.

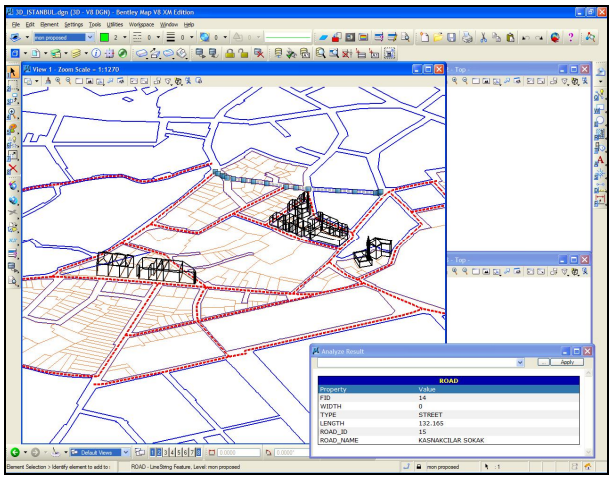


Figure 10: 2D and 3D model was uploaded to Bentley Map

3.5 Editing of 3D objects in Bentley Map

In Bentley Map, we also can edit the spatial objects using the EDIT function. After editing, we can post the new object directly to Oracle database as shown in the figure 11.

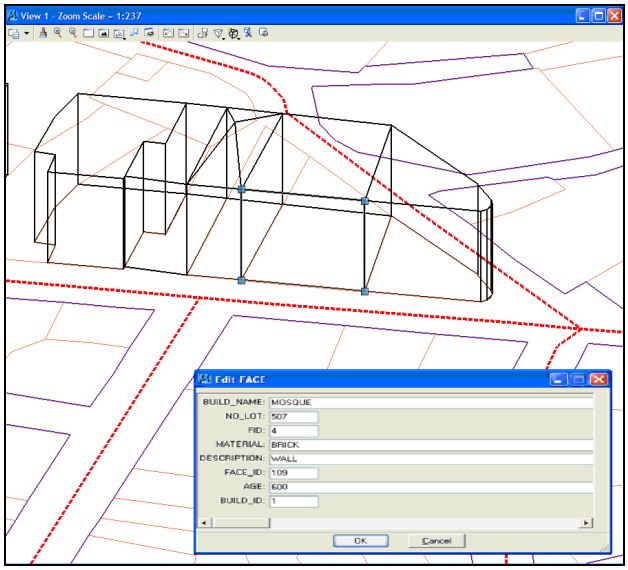


Figure 11: Edited 3D object

3.6 Query of objects in Bentley Map

There is a possibility to query spatial objects stored in Oracle database via Bentley Map. The linkage between the two systems i.e. Oracle Spatial and Bentley Map needs to be setup first to perform spatial queries. In the experiment, the registration of spatial objects are conducted using two methods, i.e. registration by faces and using multi polygons. Therefore, the query from each method will be highlighted.

In Bentley Map, the query is conducted via the visual SQL Query Builder features. Figure 12 shows the list of spatial attributes based on the selected table. The query for faces registration is conducted by selecting the faces attributes. The output of the experiment can be seen in figure 13. The same method is also applied for querying multi polygon registration. Figure 14 shows the result of the query.

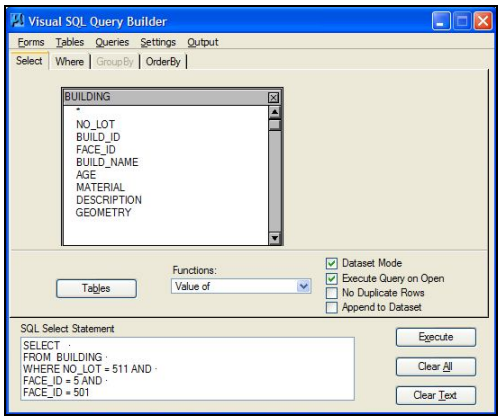


Figure 12: Visual SQL Query Builder (with list of spatial attributes)

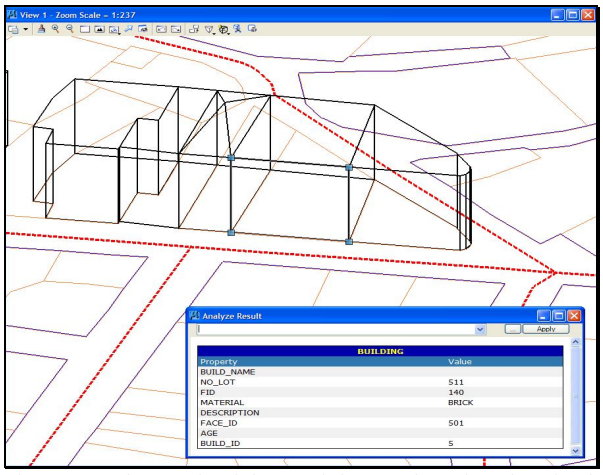


Figure 13: The result of the spatial query based on face-by-face method

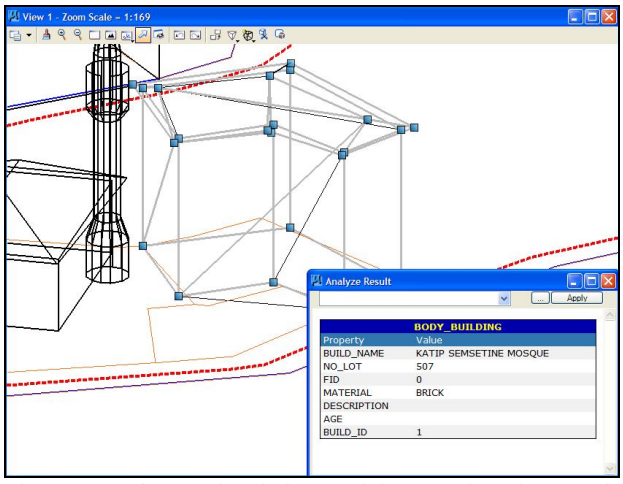


Figure 14: The result of the spatial query based on multi polygon method

4. CONCLUSION

In this paper, we have discussed how to manage 3D spatial objects on Oracle Spatial 10g and visualize in Bentley Map and exhibited valuable information related to 3D functionality currently offered. In addition visualising 3D objects stored in a DBMS is not always straightforward. With Bentley Map it is possible to visualise 3D objects rather easily, however the posting of data to the DBMS is not straight forward. The user is expected to have excellent skills in both systems, i.e. understanding the conceptual representation in Bentley Map and being aware of the implementation in storing geometrical and topological data in Oracle Spatial 10g. In the future, we would like to extend the work by addressing the automated spatial data updating and maintenance.

ACKNOWLEDGEMENT

We would like to thank to Ministry of Science, Technology and Innovations (MOSTI) for research funding (UTM RMC Vote # 79197) and also to Universiti Teknologi Malaysia for research project management.

REFERENCES

Arens, C., J.E. Stoter, and P.J.M. van Oosterom (2003), Modelling 3D spatial objects in a GeoDBMS using a 3D primitive, under review for AGILE 2003.

Batty M, Chapman D, Evans S, Haklay Mi, Kueppers S, Shiode N, Smith A, Torrens P, M, 2000, Visualizing the city: Communicating Urban Design to Planners and Decision-Makers, ISSN: 1467-1298, CASA, UCL, http://www.casa.ucl.ac.uk/visualcities.pdf.

Batty M, Dodge M, Jiang B, Smith A, 1998, GIS and Urban Design, ISSN:1467-1298 CASA, UCL, http://www.casa.ucl.ac.uk/urbandesifinal.pdf.

Bentley (2002), Bentley Map edition (2007). URL: http://www.bentley.com/products/bentley-map/

De La Losa, 1998, Toward a 3D GIS: Conception of a 3D GIS with a complete topological management, In: Proceedings of GIS PlaNET'98 Conference, Lisbon, Portugal

ESRI, 1997, Using ArcView 3D Analyst. *ESRI Publication*, Redlands, California, USA, 118. p.

Oosterom, P. v, J. Stoter, W. Quak and S. Zlatanova, 2002, The balance between geometry and topology, In: Proceedings of Spatial Data Handling, 8-12 July, Ottawa, Canada (to be published)

Oracle (2001): Oracle Spatial User's Guide and Reference Release 9.0.1 Part Number A88805-01, June 2001.

Stoter, J. and P. van Oosterom, 2002, Incorporating 3D geoobjects into a 2D geo-DBMS, Proceedings of SPRS/ACSM, 19-26 April, 2002, Washington, USA.

Varshosaz M., 2003. True Realistic 3d Models Of Buildings In Urban Areas*, International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XXXIV-5/W10.

Zlatanova, S., 2000, 3D GIS for urban development, *PhD thesis*, ITC, The Netherlands, 222 p.

Zlatanova, S., A. Abdul-Rahman, M. Pilouk, 2003, 3D GIS: Current Status and Perspectives, ISPRS and CGI (Canadian Geomatic Institution) Conference, Ottawa, Canada.