ADVANCED DATA STRUCTURES FOR SURFACE STORAGE

Karel, JANECKA[1], Michal, KARA[2]

[1]Department of Mathematics, Faculty of Applied Sciences, University of West Bohemia,
Univerzitni 22, 323 00, Pilsen, Czech Republic

kjanecka@kma.zcu.cz

[2]Georeal, Halkova 12, 301 00, Pilsen, Czech Republic

michal.kara@georeal.cz

Abstract

Nowadays the amount of spatial data is rapidly increasing. One of the reasons is the modern technique for collecting data like laser scanning etc. The demand for an efficient data processing brings in also the requirement for their effective storage. Due to the amount of data it seems useful to store them in a spatial database. Modern database management systems are more and more supporting the processing multidimensional spatial data. Most of them allow using an object-relational data model. It means that multidimensional data can be stored as an object using the appropriate abstract data type. The crucial factor for maintaining a large volume of spatial data inside a spatial database is an availability of special data structures which are suitable for such kind of data. The paper introduces the advanced data structures which are designed to support storage of surface data inside the spatial database. The focus is put on the extension of Oracle database management system for processing spatial data – Spatial as there are special data structures allowing direct storage of data representing the surface as Triangulated Irregular Network or Point Clouds. These structures are complicated in comparison with the standard Open Geospatial Consortium data types for multidimensional data. They provide the concept which seems to be useful but there are some bottlenecks for usage of real data. The paper presents the finding out about the underbellies and proposes some possible improvements.

Keywords: Data structures, spatial databases, surface storage

1. INTRODUCTION

The many initiatives have been done aiming to propose suitable data structures for a surface storage. Many earlier proposed data structures could be found e.g. in [1] or [2]. In such cases a surface is stored mostly in a proprietary file format. Some of the proposed structures could be stored in relational database management systems using the atomic data types. The figure 1 illustrates a vertex-based Triangulated Irregular Network (TIN) in which points include a clockwise list of neighbouring vertices. This structure could be simply implemented using the atomic *number* data type. Nowadays there are several important reasons why to use spatial database management systems (SDBMS) instead of files. The main ones could be that SDBMS support transactions, multiple users and editing and Structured Query Language (SQL).

The spatial database management systems can be characterized by many factors like a prize, support, speed of queries or memory consumption. Such comparisons could be found for example in [3] or [4]. Another approach can be to compare SDBMS according to the supported geometries and functions. This criterion is more related to the aim of this paper to present the advanced data structures which are suitable for direct surface storing. In this matter the most mature implementations of a spatial type system [5] were explored and only in Oracle Spatial the appropriate specialized data structures for storing surface were discovered.

Open Geospatial Consortium (OGC) delivered the standard [6] which defines Geometry object model. This model consists of many geometry classes supporting the storage of various types of geometries. One of the main classes is the *Surface* class. The *PolyhedralSurface* class is the instantiable subclass of *Surface*. It is a simple surface consisting of some number of Polygon patches or facets. OGC Geometry object model also

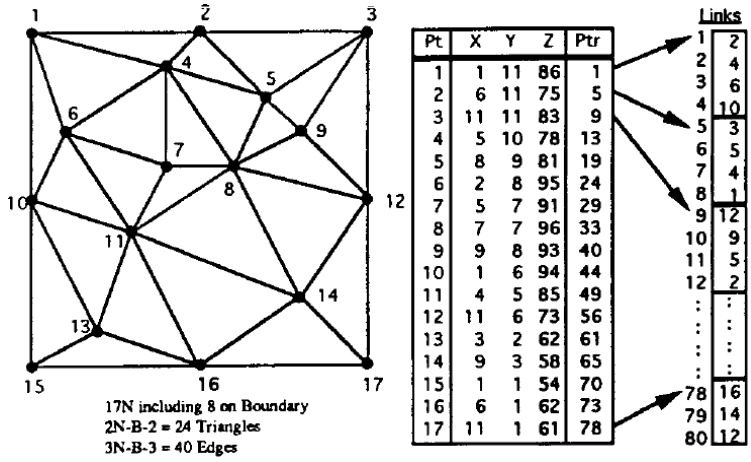
contains a class *TIN* which aims to support storage of TIN inside spatial databases. According to the OGC Geometry object model TIN could be defined as a contiguous collection of triangle patches which share common boundary segments.

For practical database implementation OGC proposed several ways how to implement Geometry object model in SQL:

- objects are implemented in SQL92 using numerical types for storage of geometric objects;
- objects are implemented in SQL92 using binary types for storage of geometric objects;
- objects are implemented in SQL92 with geometric types.

In SQL92 with geometric types the columns in a feature table are defined by feature attributes; one or more of the feature attributes will be a geometric attribute. The geometry of a feature is one of the feature attributes and is a SQL geometry type as defined in OGC Simple Feature Access standard [7]. These SQL geometry types are based upon the OGC Geometry object model.

Mostly all the main spatial database management systems support this OGC Simple Feature Access standard. Some database systems (e.g. MySQL) support only version 1.0 but most of them (e.g. PostGIS, Microsoft SQL Server, Oracle Spatial) provide user with version 1.2 allowing modelling 3D geometries. ESRI Geodatabase data format allows storing TIN as a planar graph where nodes are connected by edges to form triangles. Edges connect nodes that are close to one another. PostGIS spatial extension of PostgreSQL database management system has constructors for creating 3D geometry. TIN is modelled as a special case of polyhedral surface which is collection of adjacent triangles. Very similar situation is for Microsoft SQL Server database system. From this point of view Oracle Spatial is an example of SDBMS providing data structures and mechanisms suitable directly for TINs or point clouds.



Pt	X	Y	Z	Ptr
1	1	11	86	1
2	6	11	75	5
3	11	11	83	9
4	5	10	78	13
5	8	9	81	19
6	2	8	95	24
7	5	7	91	29
8	7	7	96	33
9	9	8	93	40
10	1	6	94	44
11	4	5	85	49
12	11	6	73	56
13	3	2	62	61
14	9	3	58	65
15	1	1	54	70
16	6	1	62	73
17	11	1	61	78

1	2
2	4
3	6
4	10
5	3
6	5
7	4
8	1
9	12
10	9
11	5
12	2
⋮	⋮
78	16
79	14
80	12

Fig. 1. TIN data structure with a clockwise list of neighbouring vertices [8].

The two advanced data structures suitable for surface storage in the spatial database (here specifically in Oracle Spatial) will be described. Both structures are based on abstract data type mechanism. The first one is designed to store surface as a point cloud. The second one supports storage of surface as a TIN.

2. ADVANCED DATA STRUCTURES

In order to add a new data type to a relational database system and to take full advantage of a new data type there are generally three different types of operations that may be specified. The first type is the primitive operations which define constants for the new type, comparisons, inserting and updating values (or how the values are actually stored). The second type of operation is the aggregate. Aggregates are operations such as the COUNT, SUM, MAX, MIN provided by the relational database management systems. Aggregate is defined to be any mapping from a single relation to a single value of a specified (possibly programmer-defined) data type. The third type of operation, the transformation, takes a relation as its parameter and returns a relation as the result [9]. Transformations could be used to convert a relation containing a surface

in one representation to a relation containing a different representation for the surface. For effective surface storing and processing in the spatial database some additional conditions must be fulfilled. The specialized data structure for a surface storage must support creation of the spatial index (e.g. R-tree) as real datasets consist of large volume of data. This is also requirement on the spatial database management system to effectively build a spatial index upon these structures.

2.1 Object-relational approach

The advanced data structures proposed for surface data and related information storing are defined using several kinds of data types. First standard database data types like NUMBER or VARCHAR are used. Due to the real spatial data bulkiness the coordinates of single points are stored in attributes of the binary large data type. For modelling other surface's attributes like minimum bounding objects the data types based on the abstract data types are used. Abstract data types extend the classical relational paradigm about the possibilities and advantages of object oriented approach to spatial data modelling. This is a hierarchal database design that is based on objects and parent-child relationships. Within object database principles and abstract data type can take on multiple forms. This allows multiple types of data to be stored in the database structure. An example of such type is SDO_GEOMETRY object data type used for storage of the minimum bounding objects. These bounding objects have the crucial role for spatial queries evaluation. The SDO_GEOMETRY object data type is defined as follows:

```
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 );
```

The *sdo_gtype* attribute indicates the type of the geometry. The possible geometries are those as specified in [7]. The *srid* attribute identifies a coordinate system. The *sdo_point* attribute stores the coordinates if a spatial object has the point geometry. For line and polygon geometries the attributes *sdo_elem_info* and *sdo_ordinates* are use. Both attributes are defined using the user defined data types. The *sdo_elem_info* attribute contains information about how to interpret coordinates stored in the *sdo_ordinates* attribute.

The data structures based on abstract data types allow an object-relational modelling spatial objects like a surface. Then the advantages of both attitudes can be used. It is possible to store the entire description of the surface in one table in one column and make relations with other objects which are stored in a spatial database.

Some practical solutions use SDO_GEOMETRY object data type only for simple storage of surface. The example of such approach could be some digital hypsography databases where however database management system serves only as a data repository [10] for storage of coordinates of points. The analysis of hypsography data and creation of surfaces are done outside the database. In case when the spatial database is used only to keep surface data (e.g. TIN) it is possible for a surface storage to use also the Well Known Text [6] or Well Known Binary [6] data formats.

2.2 Surface as a point cloud in spatial database

Current methods for collecting geographic data such as aerial and terrestrial laser scanning allow to receive a large amount of detailed data in a relatively short time. The laser scanning data are obtained in the form of point clouds. An important factor for the usage of the obtained data is their further processing. According to the scope of the data it is suitable to store and analyze such data in a spatial database management system. The number of points in a point cloud is usually thousands or millions. For such amount of data the SDO_GEOMETRY object data type is not much suitable. The reason is a restriction for the maximum of the stored numbers (vertices). Furthermore the disadvantage of using SDO_GEOMETRY could be that all points are stored in one array. It is not possible to query only a part of this array of vertices. The whole array must be loaded into a memory and due to the volume of the data this can cause performance difficulties. That requires special data structures which enable effective point clouds management directly in the database. It

means that each point in point cloud represents the point of surface. This allows generating a surface upon such stored data.

From the logical data model point of view each point cloud is stored using two tables - *Base Table* and *Block Table*. The base table is for storage of user information about a point cloud. As such it contains user defined columns. The block table serves as a place for storage of the point cloud. This point cloud placement in the block table can be done in several ways. The entire point cloud can be stored in one row of the block table. However for the real data dataset containing a huge amount of points the point cloud is stored in the more blocks (rows). Figure 2 captures this point cloud division.

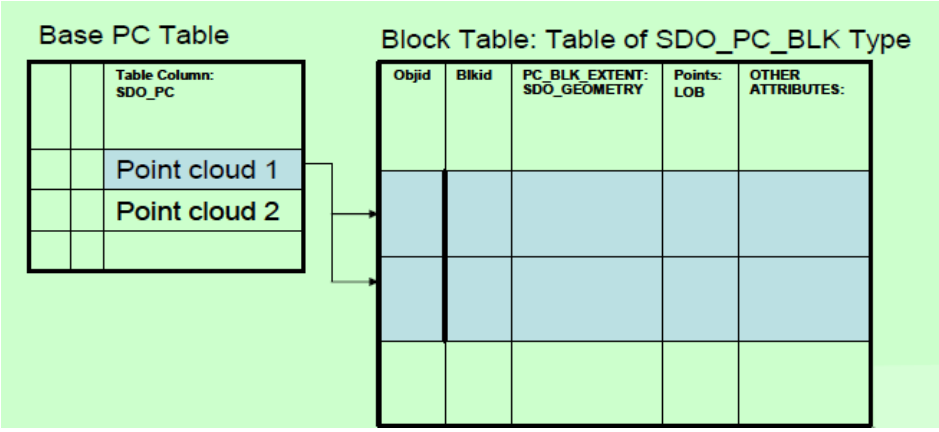


Fig. 2. The point cloud theoretically consists of the several blocks [11].

Physically the surface as a point cloud is stored and modelled using the three main data object structures:

- SDO_PC
- SDO_PC_BLK
- SDO_PC_BLK_TYPE

SDO_PC object data type

SDO_PC stores reference to smaller blocks covering the points according to their placement in a space. This allows selecting only these blocks (set of data) which are really requested. It is not needed to load all point cloud and so the time complexity won't usually be so high. The information regarding the point cloud is stored in a table which is called *Base Table* containing the SDO_PC column. The particular blocks containing the point cloud data are stored in a separate table. This table is referenced as *Block Table* and is defined using the specialized SDO_PC_BLK data type. The relation between the base table and block table is captured in figure 3. The number of points per block is given by user and depends on the size of input data set. The setting maximum number of points per block will influence the effectiveness of analysis and for example the visualization [11].

SDO_PC is further used for storage of the main characteristics of the point cloud. There the condition for a division of the points into smaller logical data blocks is stored. On principle this condition is expressed as the number of points in one block. For a spatial query evaluation this condition has a fundamental meaning. It is more effective to work with smaller units as these blocks could be spatially indexed. For this reason each logical data block is bounded by the minimum bounding rectangle (of SDO_GEOMETRY object data type). This leads to the faster spatial query evaluation. There is no possibility to set up the boundaries of block manually. It means that in case where the points will be more irregularly distributed this cannot be technically taken into consideration. For points splitting into particular blocks some heuristic value can be used. In case of Oracle Spatial the spatial database engine aims to split the points uniformly that each logical block will cover the same number of points. Furthermore attribute *pc_extent* contains object representing the spatial extent of the point cloud i.e. the minimum bounding object enclosing all objects in the point cloud. This can be used is some spatial analysis.

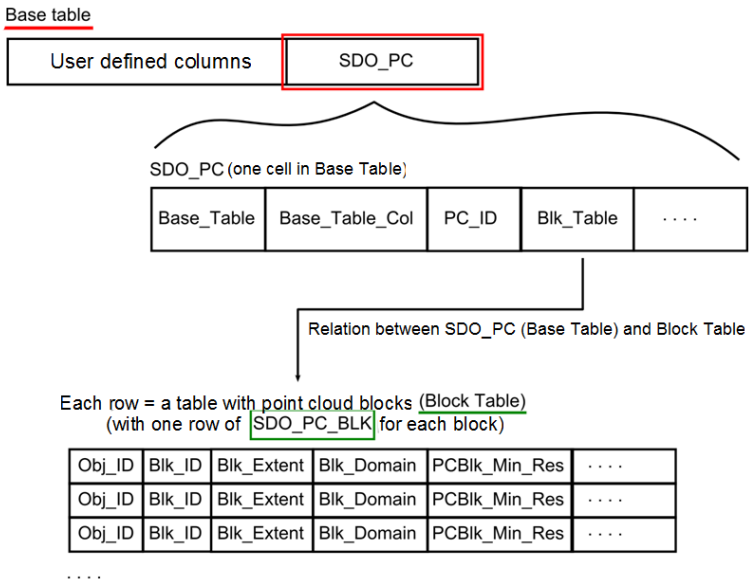


Fig. 3. The relation between the Base table and Block table [11].

SDO_PC_BLK object data type

This is the main building block of the point cloud database structure. The blocks of points and their coordinates are stored using this object data type. For each point both the identifier of the particular block and identifier of point are stored. Due to the expected number of points in real data sets the binary data type is used for storage of coordinates as can be also seen from the figure 2. This data type stores among others information about the minimum and maximum resolution level at which the block is visible in a query.

The division of points into the blocks is done in several steps:

- computation of the total number of points,
- determination of blocks which will be fully filled by points up to the maximum (threshold value),
- determination of two blocks which will be equally filled by the rest of points (see figure 4).

	NUM_POINTS
1	7115
2	7115
3	10000
4	10000
5	10000

Fig. 4. The principle of distribution of points into the blocks [12].

The default threshold value for the maximum number of points in one block is 5000. According to the size of input data it is possible to set a different value as illustrated in figure 4 where the threshold value was set on 10000 points per block. The next very important attribute for spatial queries is the *Blk_extent* attribute which stores a block spatial extent as an SDO_GEOMETRY object. The spatial index is build upon this attribute which enables faster query evaluation. This situation is captured in figure 5. If a clip operation returning the points that are within a specified query window is done an object of SDO_PC_BLK_TYPE is returned which is defined as table of SDO_PC_BLK [13].

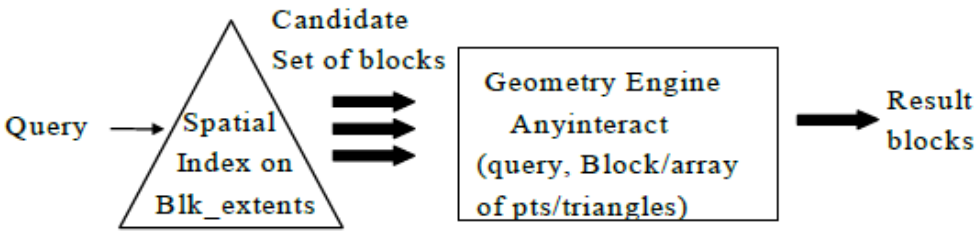


Fig. 5. A spatial query evaluation uses the attribute Blk_extent and spatial index [12].

The block is retrieved only in a case that the query window intersects the particular spatial extent and if the minimum (maximum) resolution interval of the block intersects the minimum (maximum) resolution interval of the query. The detail description of all available attributes (for SDO_PC and SDO_PC_BLK) mentions [13].

2.3 Surface as a TIN in spatial database

Oracle Spatial supports direct storage of TIN in the special data structure which is based on abstract data type mechanism. Figure 6 shows the principle of loading and processing input data into Spatial. The real datasets forming the TIN are loaded from the source flat files or some existing point tables. Furthermore nowadays the data from laserscanning are the source of very precise surface data. Spatial database management system can transform the source point (cloud) data into specialized TIN data structure. There can be done some spatial operations like clipping upon the TIN data directly on the side of spatial database.

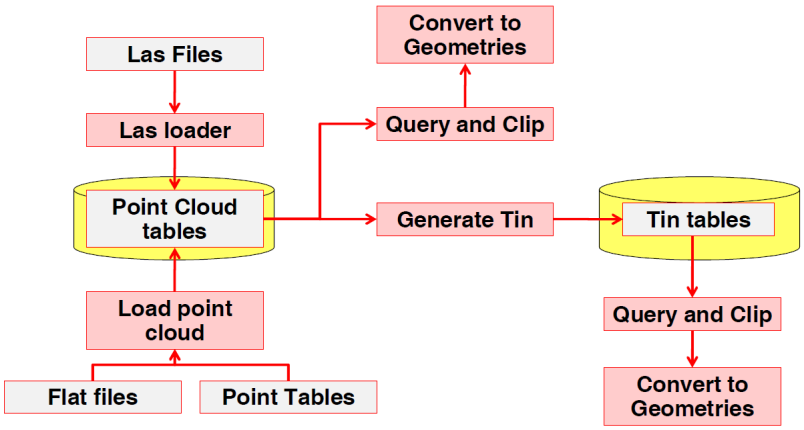


Fig. 6. The principle of processing input surface data and storage in a spatial database [14].

In Spatial a TIN is logically structured into blocks of data as figure 7 illustrates. This division allows more effective storage and retrieval of data. A spatial query on select of only some restricted area will select only particular blocks for which the query is relevant. In Spatial for a TIN creation the Delaunay triangulation algorithm is used. This algorithm calculates the most appropriate combination of triangles from the input data set. The aim at best is to create the equilateral triangles containing no other points. The input data sets are in form of a point cloud.

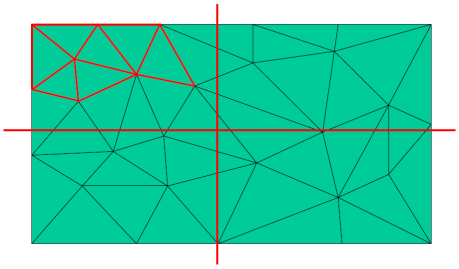


Fig. 7. TIN is logically structured into smaller blocks [15]

The TIN surface is stored and modelled using the three main data object structures:

- SDO_TIN
- SDO_TIN_BLK
- SDO_TIN_BLK_TYPE

SDO_TIN object data type

The description of a TIN is stored in a single row in a single column of object type SDO_TIN in a user-defined table (also called *base table*). This table then contains all attribute information related to the TIN. SDO_TIN holds the parameters for partition of the TIN. For spatial indexing purposes and faster spatial queries evaluation the minimum bounding object enclosing all objects in the TIN is available. The minimum bounding rectangle is of SDO_GEOMETRY object data type. The particular blocks of data are stored in a separate table modelled using SDO_TIN_BLK type. Some other attributes which are currently available in the SDO_TIN object data type are not yet supported for practical work. For example the break lines are not fully supported [13].

The relation between the table containing the SDO_TIN object and TIN block table is captured in figure 8.

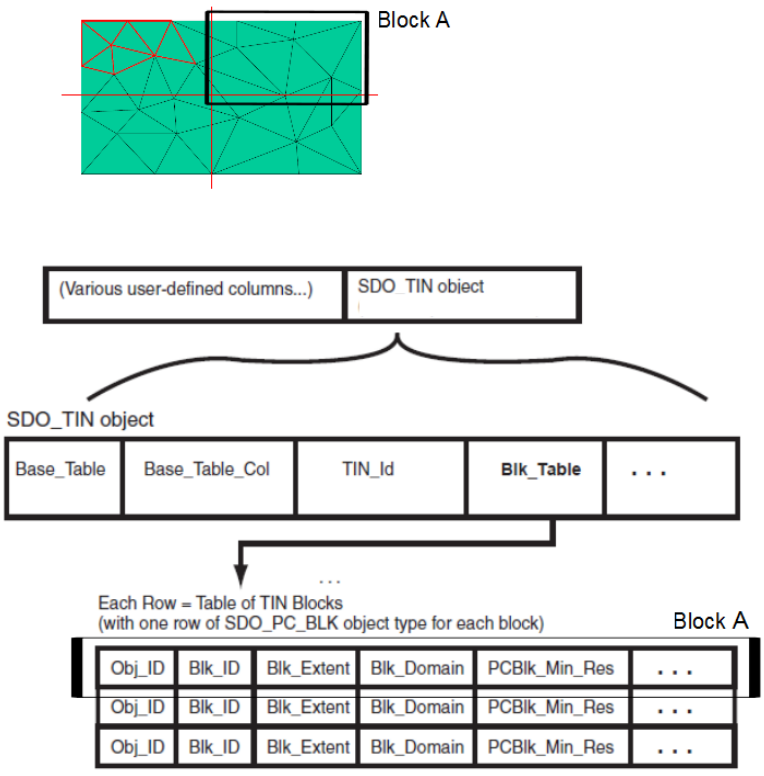


Fig. 8. Storage model for TIN data.

The attribute *tin_stop_lines* represents the stop line or lines in the TIN are the line string or multiline string stored as SDO_GEOMETRY object. Stop lines typically indicate places where the elevation lines are not continuous such as the slope from the top to the bottom of a cliff. Such regions are to be excluded from the TIN.

SDO_TIN_BLK and SDO_TIN_BLK_TYPE object data types

The TIN block table is modelled as a table with rows of the SDO_TIN_BLK type. It contains information about a spatial extent of the block, the minimum (maximum) resolution level at which block is visible in a spatial query. The spatial extent is modelled as SDO_GEOMETRY object. In the block table for each block the information about the number of points in the block is stored. The triangles shaping the TIN and points

are stored in different attributes. Points' coordinates are stored using a binary data type. This binary attribute consists of an array of points storing both identifier of particular block and point as such. The triangles are stored also as a binary sequence of data. Each vertex is specified by the identifier of the block and identifier of the point. If a clip operation is done then an object of SDO_TIN_BLK_TYPE is returned. This object data type is defined as table of SDO_TIN_BLK [13].

When the existing TIN is modified then the entire TIN must be rebuild. In case of the removing the point from the existing TIN it must be done explicitly from the base table. The standard Spatial SDO_TIN_PKG package allows transforming the TIN data into the SDO_GEOMETRY object data. Then it is easy to visualize the TIN data without any other data transformation as illustrated in figure 9.

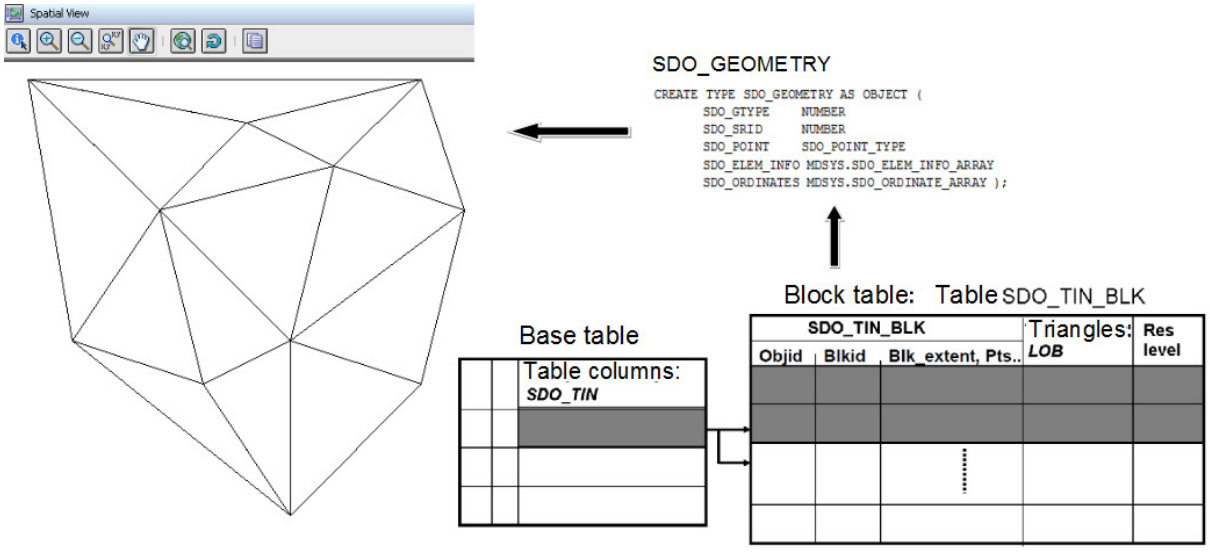


Fig. 9. Example of TIN visualization.

2.4 Further processing of the surface data in spatial database

From the spatial analysis point of view the main SDMBSs support mainly such data types like SDO_GEOMETRY. The reason is that it is relatively easy to create a function working with features which are in spatial database modelled as objects. However if the surface is stored using the appropriate data structure the best for spatial queries (analyses) is if a spatial database would support operations also upon these advanced data structures. Due to the novelty and particularity of the presented data structures there is relatively weak build-in support in Spatial. For spatial analyse which should be done directly in a spatial database the possible solution seems to be either to use the implicit function transforming the surface data or its part into the data type like SDO_GEOMETRY or using a function-based mechanism. Then it is possible to apply all available standardized functions working with this object data type as illustrates the following example showing the selection of all blocks within a specified distance from the reference object.

```
SELECT  blk_id, obj_id
FROM    pc_blktab
WHERE   SDO_WITHIN_DISTANCE(
          VRAT_GEOM(points,num_points),   --function returning the geometry description
          SDO_GEOMETRY(2003, NULL, NULL,
            SDO_ELEM_INFO_ARRAY(1,1003,3),
            SDO_ORDINATE_ARRAY(-812783,-1079712,-812782,-1079712)),
        'distance=100 min_resolution=10') = 'TRUE';
```

The main advantage of using a function-based index is that the data remain store in the specialized data structure and are ad-hoc materialized and hence no other space for storage of the surface data is needed. Table 1 gives an overview about limitations for storing.

Table 1. Limitations for particular object types.

Object Type	Attribute	Limitation
"classic" SDO_GEOMETRY	SDO_ORDINATE_ARRAY	349 525 of 3D points
"uber" SDO_GEOMETRY	SDO_ORDINATE_ARRAY	333 333 of 3D points
SDO_PC_BLK	POINTS	BLOB - 32 bytes for each point
SDO_TIN_BLK	SDO_POINT	for point cloud data - 12 MB (for Oracle Database Release 11.1.0.7 or later); approximately 393k of 3D points
SDI_TIN_BLK	TRIANGLES	BLOB - limitation is not documented

3. CONCLUSION

The trend of storage of large amounts of data into spatial databases is evident. If the spatial database should serve not only as data storage then there is the need for specialized data structures. These structures should allow to model, store and process the surface data effectively. The structure should provide a possibility to store also user-defined information. If the focus is put on modern spatial database management systems then the more specialized data structures based on abstract data types can be found in Spatial. As the structures implementations are relatively new there is a lack of experience with real data but some future improvements are obvious. The detailed information about how to use the presented data structures implemented in Oracle Spatial for the storage and processing of the real surface data can be found mainly in [11]. These advanced data structures are fully integrated into the spatial database management system and support spatial indexing and spatial analysis as well.

The concept of building such structures upon abstract data types seems to be suitable for further development of both data structures and related functions as computation of inclination of slope etc. The weakness for the usage of the current TIN data structure implementation can be no support for direct data visualization (the same for the point cloud data structure). In case that some spatial analysis will be done on the side of spatial database no presence of a native visualization tool could be limiting but it is only a minor issue. As some points may lay directly on the boundary between two blocks the data could be stored redundantly. Some triangles can also belong to more than one block as illustrated in figure 7.

For the point cloud data structure the storage of additional non-spatial attributes related to the points will require an extension of this data structure. At this moment it is possible to store only information related to the point cloud as a whole in user defined attributes (columns). In case that it will be necessary to update the point cloud then this new actualized point cloud must be created from the beginning. For example there is no default function for adding one or several new points into the existing point cloud. There is also no exact algorithm about how to set the threshold value for maximum number of points in one block. The result from the testing is that this should be done according to the size of input data with aim to receive acceptable performance results for spatial analysis.

The further improvement and development of the proposed data structures should focus on an additional support for surface modification and analyses. This fact should be taken into account for example by architects who are responsible for creation of geoinformation infrastructure or GIS specialists who would like to use the spatial database in their work flow processing the surface data. The procedural extension of the Structure Query Language allows to propose and implement own functions and procedures which will use the built-in spatial functions. Due to the fact that surface data are stored directly in spatial database it will enable not only to create a surface and storing it in the appropriate advanced data structure but also e.g. to make the surface generalization and further spatial analysis directly by means of SDBMS.

REFERENCES

[1] Kidner, D.B. (1991) Digital Terrain Models for Radio Path Loss Calculations, Unpublished Ph.D. Thesis, Department of Mathematics & Computer Studies, The Polytechnic of Wales, U.K., November, 269 pages.

[2] Packer, T.K., Fowler, R. J., Little, J.J. and Mark, D.M. (1978) The Triangulated Irregular Network, Proceedings of the Digital Terrain Models (DTM) Symposium, ASP/ACSM, May 9-11, pp.516-540.

[3] Compare SQL Server 2008 R2, Oracle 11G R2, PostgreSQL/PostGIS 1.5 Spatial Features http://www.bostongis.com/PrinterFriendly.aspx?content_name=sqlserver2008r2_oracle11gr2_postgis15_compare
Cited: 03/12/2011

[4] Cross Compare SQL Server 2008 Spatial, PostgreSQL/PostGIS 1.3-1.4, MySQL 5-6 http://www.bostongis.com/PrinterFriendly.aspx?content_name=sqlserver2008_postgis_mysql_compare
Cited: 03/12/2011

[5] Greener, S. Oracle Spatial for PostGIS Users – Understand, Isolate and Migrate. The SpatialDB Advisor. http://download.osgeo.org/osgeo/foss4g/2009/SPREP/2Thu/Parkside%20Auditorium/1300/
Cited: 03/12/2011

[6] OGC 06-103r4. OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture. Open Geospatial Consortium Inc.

[7] OGC 06-104r4. OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 2: SQL option. Open Geospatial Consortium Inc.

[8] Kidner, D. B. and Smith D. H. (1993) Data structures for terrain modelling and ground cover data. Proceedings of Terrain Modelling and Ground Cover Data for Propagation Studies.

[9] Oosborn, S. L. and Heaven, T. E. (1986) The Design of a Relational Database System with Abstract Data Types for Domains. ACM Transactions on Database Systems. Volume 11 Issue 3.

[10] Potrebova, L. Technology solution of the national hypsography databases. University of West Bohemia. Pilsen. 2011.

[11] Kara, M. Storage and analyses of the point cloud in Oracle Spatial. University of West Bohemia. Pilsen. 2011.

[12] Ravada, S.; Horhammer, M.; Kazar, B. Point Cloud. Storage, Loading, and Visualization http://www.cigi.illinois.edu/cybergis/docs/Kazar_Position_Paper.pdf
Cited: 15/04/2011

[13] Murray, Ch. Oracle Spatial Developer's Guide 11g Release 1. Oracle, 2009 http://download.oracle.com/docs/cd/B28359_01/appdev.111/b28400.pdf
Cited: 03/2010

[14] Kara, M. TIN in Oracle Spatial. University of West Bohemia. Pilsen. 2010.

[15] Query Processing In 3-D Spatial Databases. 2008 Http://3dgeoinfo.Uos.Ac.Kr/Data/Presentation/All/Session02/1_3d_Query_Processing_3dgeoinfo08_Presentation.Pdf
Cited: 10/12/2009