

FREEFORM CURVES AND SURFACES IN DBMS: A STEP FORWARD IN SPATIAL DATA INTEGRATION

S. Zlatanova¹, S. Pu² and W.F. Bronsvoort³

¹GIS, OTB, Delft University of Technology, Delft, The Netherlands
s.zlatanova@tudelft.nl

²ITC, Enschede, The Netherlands
spu@itc.nl

³Computer Graphics and CAD/CAM, Delft University of Technology, Delft, The Netherlands
W.F.Bronsvoort@tudelft.nl

Commission IV/8

KEY WORDS: Databases, CAD, GIS, Data structures, Retrieval, Modelling, NURBS

ABSTRACT:

Spatial data integration is of great importance for many applications and particularly for emergency response. Besides the variety of challenges in 2D, increasing attention is given to integration of CAD and 3D GIS models. Many indoor models (needed for localisation and evacuation of people) or geological formations (needed for earthquakes or landslides) exist in CAD systems, but are often difficult (or impossible) to integrate with GIS models. Nowadays, a process of converging functionality is observed, but still many challenges remain. Amongst all these are the supported primitives. CAD software supports a broad range of primitives such as cone, sphere, cylinder and free-form curves and surfaces (NURBS, B-Splines, Bezier), which are not present in the GIS world. Open Geospatial Consortium has recognized the importance of freeform curves and surfaces and has included them in the Abstract Specifications, but no GIS software supports them. This paper presents our design of six new freeform data types to be considered for SQL Implementation Specifications.

1 INTRODUCTION

GIS and CAD systems used to differ significantly and this is logical: the two types of software were designed for different purposes. Initially, the intention of CAD systems was to provide 3D tools for design of relatively small models (constructions, industrial parts, cars, etc.) in local coordinates systems. Since this was design software, a lot of emphasis was given to editing tools and effective 3D visualization. In contrast, GIS was designed to represent the real world, and more specifically all the tasks that used to be performed on paper maps. GIS, therefore, was developed to maintain points, lines and polygons with geographic coordinates and corresponding attributes, and provide specific spatial analysis (very much application-oriented). Many authors have discussed the similarities and differences between CAD and GIS and suggested different way to integrate them (Shepherd 1990, Schutzberg 1995, Plümer 2004, Oosterom et al, 2006). Technology- or application-driven, many developments have been contributing to seamless exchange of data between CAD and GIS, which is not yet completely successful. Despite promising research results (Arens et al 2005, Penninga 2005), the general problem, i.e. lack of 3D primitives in GIS environments, remains. This paper presents an approach of linking the two domains, by providing 3D complex data types at database level.

The developments in this paper follow the formal approach of specifying geometry used in GIS domain, i.e. the Open Geospatial Consortium (OGC). Formalization initiatives exist also in the CAD domain (e.g. ISO 10303). However, this

standard concentrates more on product-related data, covering life-cycle aspects in design and manufacturing.

The formal geometry semantics for real-world objects (or spatial features in OGC terms) is metrically and topologically described in the OGC Abstract Specifications, (OGC, 2001), which are identical with ISO 19107. The geometry of spatial features is described by the basic class `GM_Object`, which is a combination of a geometry and a coordinate reference system. The Abstract Specifications do have an extended support of primitives as they are used in CAD, including freeform shapes, i.e. Bézier, B-spline, Cubic-spline, Polynomial spline and NURBS¹. In this paper we present research that goes one step further. We design a data type for the SQL Implementation Specifications (i.e. for an implementation in DBMS).

The rest of the paper is organised as follows. Section 2 gives a brief definition of freeform shapes. Section 3 presents the conceptual design of the data types. Section 4 presents some implementations results. The last section concludes on the obtained results and recommends further developments.

2 BÉZIER, B-SPLINE AND NURBS CURVES AND SURFACES

There are several methods to represent freeform curves and surfaces. Bézier, B-spline and NURBS methods are among the

¹ This work has discovered an error in the description of NURBS in the Abstract Specifications. The editor J. Herring has been notified and corresponding corrections are to be introduced.

most commonly used in practice, and are therefore considered here too. Bézier, B-spline and NURBS curves and surfaces are all represented by parametric functions (Piegl and Tiller 1997). Parametric functions have several advantages over implicit functions, i.e. functions of the form $f(x,y,z)=0$. Some of the most important advantages are:

- They have more degrees of freedom to model shapes than implicit functions have.
- Points on a curve or surface can be evaluated reasonably fast by numerically stable and accurate algorithms.

The simplest of the three methods to represent a freeform curve discussed here is the Bézier curve. Its shape is basically defined by a sequence of $n+1$ control points P_i ($i=0..n$) in 3D space. A Bézier surface is, similarly, defined by a grid of $(n+1)*(m+1)$ control points $P_{i,j}$ ($i=0..n, j=0..m$). One of the major problems with Bézier curves and surfaces is that one usually wants to keep the degree(s) low, i.e. preferably not higher than 3 or 4, and therefore one has to model complex shapes by a composition of several curves or surfaces. This, in turn, requires specific configurations of control points to guarantee a certain order of continuity.

B-spline curves and surfaces, which are generalizations of Bézier curves and surfaces, do not have this problem, because here the degree can be defined independently from the number of control points, and continuity of the curve or surface is realized automatically. A B-spline curve of degree p , or order $k=p+1$, is defined by a sequence of $n+1$ control points P_i ($i=0..n$) in 3D space, and a knot vector of $n+k+1$ knots. It is a piecewise polynomial curve.

A B-spline surface of degrees p and q , or orders $k=p+1$ and $l=q+1$, in u respectively v is, similarly, defined by a grid of $(n+1)*(m+1)$ control points $P_{i,j}$ ($i=0..n, j=0..m$), and two knot vectors U and V . The choice of the knot vectors has quite some influence on the resulting shapes, including the degree of continuity at the knots. In particular, one can take uniform or non-uniform knot vectors; in a uniform knot vector, all knots are chosen at equal intervals. See Piegl and Tiller 1997 for more details.

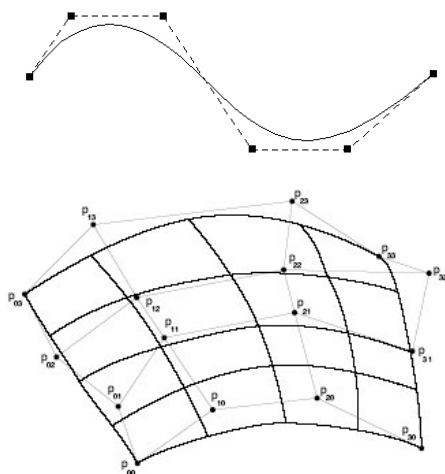


Figure 1: NURBS curves (above) and surface (below)

Though Bézier and B-splines are widely used representations, the most popular method for representing freeform shapes is nowadays the NURBS method (Figure 1). NURBS are a generalization of B-splines. The main difference between

NURBS and B-splines is that the control points of a NURBS curve or surface each have a weight, which determines how much the control point contributes to the curve or surface. This gives extra degrees of freedom for modelling curves and surfaces. The most important properties of NURBS curves are:

- A NURBS curve is a piecewise rational polynomial curve, and has the same continuity conditions at knots as a B-spline curve.
- NURBS curves are projective invariant, i.e. one can apply affine and projective transformations by applying these to the control points.
- NURBS curves can exactly represent conic sections, such as circles and ellipses.
- NURBS curves are, just like B-splines curves, locally modifiable and contained within the convex hull of their control points.

Similar properties are valid for NURBS surfaces. In particular, these can exactly represent quadratic surfaces, such as a spherical surface.

Altogether, the NURBS method is one of the most powerful methods for representing freeform curves and surfaces. NURBS have been included in many geometric standards, and are supported by many mainstream CAD systems.

3 CONCEPTUAL DESIGN

The most important information from the previous section for modelling a new freeform data type is on the parameters that have to be maintained for the data type. It is clear that NURBS curves require the largest number of parameters, i.e. *control points, knot vector, degree and weights*. Bézier curves require only *control points and degree*. B-spline curves need a *knot vector* in addition to the parameters of a Bézier shape.

In fact, all the parameters discussed in the previous section for Bézier and B-spline curves and surfaces can be defined as attributes in the classes. Bézier and B-spline are to be defined as instances of `GM_BsplineCurve` (an abstract class for curves and surfaces). A Bézier curve can be described as a B-spline curve in which the knot vector contains a special sequence of values. There are two other optional attributes, i.e. *curveForm* and *isPolynomial* mentioned in the OGC Abstract Specifications, which are used to indicate the type of curve to be approximated and whether a curve is polynomial.

3.1 Bézier, B-spline and NURBS data types

The conceptual model has been created with respect to the mathematical definitions, considering the OGC Abstract Specifications and OGC Implementation specifications for SQL (Figure 2). Optionally, the definition of the new NURBS data type can strictly follow the existing OGC formalism by creating a new class for NURBS. NURBS can be considered a generalization of B-spline and Bézier and, consequently, the NURBS class can be defined as a super class of `GM_SplineCurve`. However, from implementation point of view (and considering Simple Feature Specifications for SQL), this approach is not that practical due to the following considerations:

- Being the most complex type, NURBS require more parameters to be defined as attributes, which leads to a complicated hierarchy between the three curves. It will mean maintaining empty values for some parameters.

- Most of the algorithms (important for developing functions and operations) are different for Bézier, B-spline and NURBS curves and surfaces. A typical example is computing the offset of NURBS curves and surfaces.

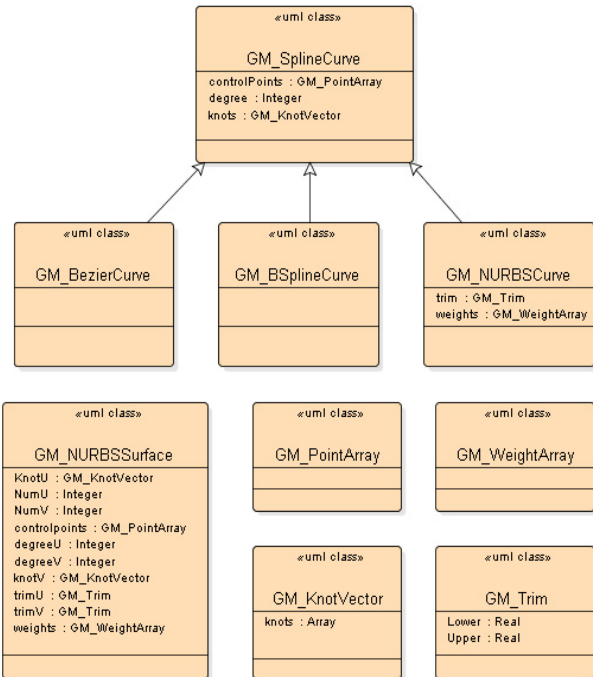


Figure 2: UML schema of four implemented data types (GM_BsplineSurface and GM_BézierSurface are not shown)

Therefore six new classes have been created as sub-classes of GM_SplineCurve: three for the curves and three for the surfaces. GM_SplineCurve is the super-class for GM_BézierCurve, GM_BSplineCurve and GM_NURBSCurve. Control points, knots and degree are attributes of GM_SplineCurve, and they are inherited by the other three curves. GM_KnotVector is the class for knot vector, which is used in GM_SplineCurve, GM_NURBSCurve and GM_NURBSSurface. GM_PointArray and GM_WeightArray are simple array types, which enumerate real values and are used to represent correspondingly control points and weight values in NURBS. GM_Trim represents the trim values for GM_NURBSCurve and GM_NURBSSurface; trim values are used to represent a part of a curve or surface.

3.2 Functions on freeform data types

Besides the data types, a set of operations on freeform shapes has to be also available, but a formalism for this has still not been developed within the GIS domain. OGC Abstract Specifications discuss only derived topological operations and suggest using three different formalisms for it (Boolean, Egenhofer and Clementini operators). OGC Implementation Specifications refer to an enlarged set of operators and functions, including metric operators (distance, area, etc.), proximity operators (within distance) and even creation of aggregates, but they are specified only for simple features. Assuming that freeform curves and surfaces should be dealt with as any other geometry data type (although more complex), operators similar to those for simple features have to be provided. This is to say that operations for validation, detecting topological relationships, metric computations (length, distance, area, etc.), proximity (distance between two features, objects within distance, etc.), operations creating new geometries

(intersection, difference, union, buffer, etc.) might be considered.

Other operations that are important for freeform curves and surfaces are rotation, translation and scaling. Such functions are needed for integrating newly designed constructions (usually in local coordinate systems) with real world models (using geographical coordinate systems). Conversion functions between Bézier, B-spline and NURBS curves will be appropriate as well, because different CAD applications support different freeform data types.

The question which functions should be available at DBMS level is very relevant (Zlatanova and Stoter 2006). On the one hand, DBMS are not designed for heavy computations. On the other hand, various complex algorithms are already available in the CAD domain. In fact, the same strategy should be followed for freeform shapes as for simple features: generic functions have to be available in DBMS, and complex computations at application level. For example, an important operation is curve evaluation, but the algorithms are complex and they are available in nearly all CAD systems (AutoCAD, MicroStation).

The functions developed within this research are relatively simple and aiming at demonstrating the use of the developed data types. They include functions for: validation, simple transformations, conversion between different freeform shapes, computing distance between two freeform curves, and retrieving parameters of NURBS such as knots, control points and their number.

The validation function for NURBS is derived from the mathematical description of the curves and surfaces and checks whether all needed parameters are present, i.e. coordinates for control points, degree value, weight values and knot vector. Moreover the following conditions should also be satisfied: degree > 1, number of control points > 3, degree = number of knots - number of control points - 1, number of weight values is equal to the number of control points, each weight value > 0, knot vector is non-decreasing and has more than 1 knot, and upper trimming value > lower trimming values.

4 IMPLEMENTATION IN ORACLE SPATIAL

We have selected to implement the new geometries as individual data types as recommended by OCC specifications, and therefore outside the SDO_GEOMETRY model. There are two major advantages of mapping the conceptual model into separate freeform data types: 1) the data types are very clear and explicitly defined, and 2) little redundant information is stored since every data type has its own attributes. Adapted data types can be easily inherited from existing prototypes, and functions on prototype types will be also operational for inherited types. This object-oriented mechanism is quite important for developing new applications. But there are also a few disadvantages. For example, different kinds of geometries have to be stored in separate table columns. This may lead to a practical inconvenience. A model normally consists of different geometry types, and it is desirable to have all the information stored in the same table column (as is the case with SDO_GEOMETRY). Furthermore, certain programming codes are required to create new data types. For example, Java or C or PL/SQL code is required in Oracle Spatial for user-defined data types.

4.1 Implementation

New data types can be designed in Oracle using natively supported data types, such as object types, REFs, VARRAYs, and Nested tables. Here we are only interested in object types

and VARRAYs. The user-defined data types in Oracle can be declared using the SQL statement CREATE TYPE. The implementation of the declaration can be PL/SQL, Java or C++. In our implementation, Java has been selected due to the good support of Oracle Spatial. The overall procedure for creating data types using Java can be subdivided into three main steps: 1) Java class creation, 2) loading of the classes in Oracle spatial, and 3) declaring the data type in Oracle using the SQL statement CREATE TYPE. The mapping between Java classes and Oracle Spatial data types is quite straightforward: Java classes map to Oracle spatial data types as Java attributes map to data type attributes. Further details on the Java implementation can be found in Pu (2005).

The functions on freeform data types have been implemented either as SQL methods (unary operations) or as SQL functions (binary operations). The implemented unary operations are: Num, NumV, NumU (number control points of curve and surface in *u* and *v* directions), Degree, DegreeV, DegreeU (degree of curve and surface in *u* and *v* directions), Cpoints (list of control points), IsClosed, Centroid (mass point of control points), ConvexHull (convex hull of control points) and BoundingBox (MinMax Bounding Box of control points). The transformation functions are Rotate, Translate and Scale as the computations are with respect to the control points. The implemented binary functions are: DistanceS2S (distance between surfaces), DistanceC2C (distance between two curves) and AnyIntersect. The distance functions are implemented as the distance between the centroids of the geometries. The function AnyIntersect checks whether two freeform geometries may intersect with each other. This operation is implemented by checking whether their convex hulls of the control points intersect (Figure 3). If convex hulls do not intersect, then the two freeform geometries do not intersect either; otherwise they might or might not intersect.

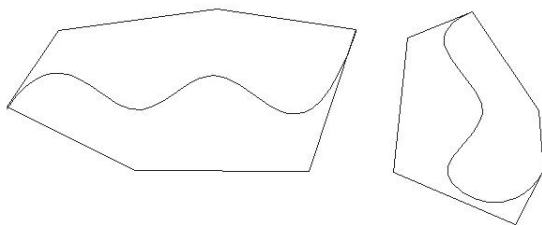


Figure 3: Example of AnyIntersect: the convex hull of the curves is used

4.2 SQL examples with created freeform data types

Following we give several examples to show how freeform data types in Oracle can be manipulated using SQL commands. The following SQL statements show the components of the GM_NURBSCurve data type

```
SQL> desc GM_NURBSCurve
GM_NURBSCurve extends GM_SPLINECURVE
```

Name	Type
DEGREE	NUMBER
CONTROLPOINTS	GM_POINTARRAY
KNOTS	GM_KNOTVECTOR
WEIGHTS	GM_WEIGHTARRAY
TRIM	GM_TRIM

A table TEST1 is created with two columns ID and GEOM. The column GEOM is intended for GM_BsplineCurve.

```
SQL> create table test1 (id number,geom
GM_BsplineCurve);
```

Table created.

A B-spline curve with ID=3, degree equal to 4, 5 control points and uniform knot vector is inserted in the table TEST1;

```
SQL> insert into test
values(3,GM_BsplineCurve(4,GM_PointArray(1,2,10,
1,2,3,9,2,3,5,7,6,4,4,4,9,0,4)
,GM_KnotVector(Vector(0,0,0,0,0,0.5,1,1,1,1),N
ULL),NULL));
1 row created.
```

```
SQL> select * from test1;
ID
-----
3
GM_BSPLINECURVE(4,
GM_POINTARRAY(1,2,10,1,2,3,9,2,3,5,7,6,4,4,4,9,0
,4),
GM_KNOTVECTOR(VECTOR(0,0,0,0,0,0.5,1,1,1,1),
NULL), NULL)
```

Let us now create a second table TEST2 with 2 columns. The column GEOM is of the freeform surface type: GM_NURBSSurface.

```
SQL> create table test2 (id number, geom
GM_NURBSSurface);
Table created.
```

A NURBS surface is inserted into the table TEST2. The parameters of this NURBS surface are: 3 control points in *u* direction, 5 control points in *v* direction; degree 2 in *u* direction; degree 2 in *v* direction; knot vector in both *u* and *v* directions; 15 weight values; no trimming value.

```
SQL> insert into test2 values(1,
GM_NURBSSURFACE(3, 5, 2, 2,
GM_POINTARRAY(5.1469375, 1.83903125, 1.744375,
5.14721875,2.24558333, 1.722375, 5.1475, 2.421,
1.98245833, 5.40390625,1.83903125,1.744375 ,
5.371125, 2.2166875, 1.759125, 5.33834375,
2.421, 1.9854375, 5.660875,1.83903125, 1.744375,
5.5950 3125, 2.18780208, 1.795875, 5.5291875,
2.421, 1.98841667, 5.910875,1.83903125,
1.744375, 5.81545833 , 2.15890625, 1.832625,
5.72003125, 2.421, 1.99139583, 5.910875,
1.83903125, 1.994375,5.910875, 2.1 3002083,
1.994375, 5.910875, 2.421, 1.994375),
GM_WEIGHTARRAY(1, 1, 1, 1, 1, 1, 1, 1, 1,
.707106781, .853553391,1, 1, 1, 1),
GM_KNOTVECTOR(VECTOR(0, 0, 0, 1, 1, 1), NULL),
GM_KNOTVECTOR(VECTOR(0, 0, 0, .5, .5, 1, 1, 1),
NULL),NULL,NULL));
1 row created.
```

The implemented unary operation NumU supplies the number of control points in the *u* direction:

```
SQL> select a.geom.NumU from test2 a;
geom.NUMU
-----
3
```

The implemented unary operation DegreeV applied to the same table gives the ID of the NURBSsurface(s) with degree 2 in the *v* directions:

```
SQL> select a.id from test2 a where
a.geom.degreeV=2;
ID
-----
1
```

The following SQL statement checks whether the B-spline curve from the table TEST1 is valid:

```
SQL> select a.geom.validation() from test1 a;
A.GEOM.VALIDATION()
-----
1
```

Value 1 means that the curve is valid (with respect to the rules defined in Section 3.2).

Let us now demonstrate the validation function for three NURBS curves.

```
SQL> create table test3 (id number, geom
GM_NURBSCurves);
Table created.
```

```
SQL> insert into test
values (1,GM_NURBSCurve(6,Null,Null,Null,NULL));
1 row created.
```

```
SQL> insert into test
values (2,GM_NURBSCurve(4,GM_PointArray(1,2,3,1,2
,3,9,2,3,5,7,6,4,4,4,9,0,4),
GM_KnotVector(Vector(0,1),NULL),GM_WeightArray(1
,1,1,1),NULL));
1 row created.
```

```
SQL> insert into test
values (3,GM_NURBSCurve(3,GM_PointArray(1,2,10,1,
2,3,9,2,3,5,7,6,4,4,4,9,0,4),G
M_KnotVector(Vector(0,0,0,0,0,0.5,1,1,1,1,1),NULL
),GM_WeightArray(1,1,1,1,1,1
,NULL));
```

```
SQL> select id, a.geom.validation() from test a;
```

```
ID A.GEOM.VALIDATION()
-----
1 0
2 0
3 0
```

The first NURBS curve is invalid (return value 0) because an insufficient number of parameters is stored; the second is invalid because the number of weights is not equal to the number of control points ((length of GM PointArray)/3); the curve with number 3 is invalid because the third geometry rule (relation between degree, number of control points and number of knots) is violated.

4.3 Visualisation in CAD software

The freeform data types in Oracle Spatial can also be accessed by CAD software. The way to do this (when no native support is available yet) is to use an application that ‘translates’ the internal CAD representation into the developed data types (and vice versa) via a database connectivity bridge. We have experimented with Microstation and AutoCAD. In the case of Microstation, the tools used were Java Microstation Developing Language (JMDL) (Bentley, 2005) and Java Database Connectivity (JDBC) Bridge (Oracle, 2003a); for AutoCAD these were ObjectARX (C++ based language), (AutoCAD, 2005) and Open Database Connectivity (ODBC) bridge (Oracle

2003b). Taking Microstation as example, a freeform model can be imported from Microstation to Oracle Spatial as follows:

1. A JMDL program checks all the shapes in the Microstation model, and constructs an instance of the corresponding freeform class (if a freeform geometry is found).
2. Instances of freeform classes in JMDL are reorganized according to the format of freeform types in Oracle, and inserted into Oracle using JDBC Bridge.

Importing data types from Oracle Spatial to Microstation is just the other way around. The procedure is similar for AutoCAD and Oracle Spatial. Further details on implementation can again be found in Pu (2005).



Figure 4: The Dutch Flower festival, Haarlemmermeer 2002

The data types have been tested with the building of the Netherlands pavilion, built for the Flower Festival 2002 in Harlemmermeer (Figure 4). The building was originally designed in Maya, using 6 large NURBS surfaces, several other smaller ones, and a number of planes. The model was imported in Microstation using the IGES file format. The data type used for storage of the geometries is GM_NURBSSurface. The planes are represented by the SDO_GEOMETRY type polygon.

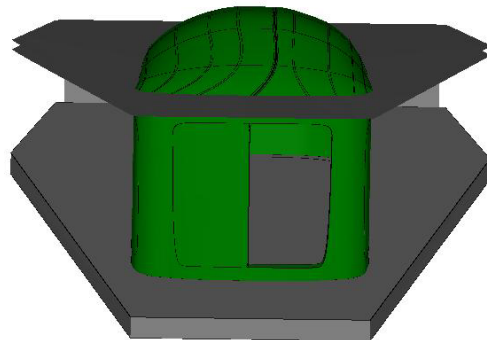


Figure 5: The 3D model retrieved in Microstaion from Oracle Spatial.

As mentioned before, the new data types cannot be stored in the same column containing the SDO_GEOMETRY types. Therefore the model was stored in two different tables: SIMPLE_GEOM and NURBS_GEOM. The polygons and complexes of them are stored in the table SIMPLE_GEOM and the surfaces in another table NURBS_GEOM. The geometries of both tables were successfully retrieved and visualised in Microstation and AutoCAD (Figure 5).

5 CONCLUSIONS AND RECOMMENDATIONS

We have shown that NURBS is a very general representation of freeform shapes, and we recommend it to be considered for including in the Implementation specifications of OGC. NURBS, Bézier and B-splines are already recognized in the CAD domain as flexible geometries for modelling new designs. These geometries can also play an important role in GIS for maintaining complex surfaces obtained from reconstruction of real-world objects.

Tests have convincingly demonstrated that appropriate data types for efficient management of freeform curves and surfaces can be created at DBMS level. The new data types have been prototyped for Oracle Spatial, but outside the Oracle Spatial SDO_GEOMETRY model, which means that they can be readily used for any spatial DBMS (PostGIS, MySQL, Informix, etc.). The design is compliant with OGC Abstract Specifications.

This research is only the first step toward managing freeform data types in a spatial DBMS and GIS. Many issues have to be further investigated. For example, the validation rules for freeform curves and surfaces have to be further specified. The current implementation of validation functions is derived straightforward from the mathematical definitions.

Further research is needed to determine relevant functions for support at DBMS level. The implemented set of prototype functions is relatively basic and not sufficient. For example, the implemented function *AnyIntersect* uses the convex hulls of the control points of two shapes to investigate the intersection. More accurate conclusions on the intersection of the shapes would require exhaustive computations. A separate data type for error messages would be of great help since the freeform shapes are much more complex than the simple features. It can be included as an error number attribute and an error message attribute within the data type.

Although NURBS can represent conic sections (circle, ellipse, hyperbola, parabola), still separate data types have to be designed to represent these simple shapes in their simplest form. The parameters needed to store, for example, a circle as a NURBS shape are considerable. However, the properties of NURBS still can be used to perform operations only on NURBS data types.

This research did not consider indexing of freeform shapes. As it is well known, a spatial index can make spatial queries much more efficient. The current spatial indexing mechanism of DBMS works on existing simple geometries and cannot be used for prototype data types. Further investigations are needed to develop an indexing mechanism for freeform geometries.

The data types do not yet resolve modelling of freeform shapes in GIS applications. Presently the only possibility is approximation of freeform shapes with simple features (sets of lines and polygons) only for visualization with 3D GIS models. Alternatively, GIS models stored in DBMS can be integrated with CAD freeform models and further explored (and modelled) in CAD applications. In this manner, a first real integration of typical CAD shapes with GIS shapes can be realized.

ACKNOWLEDGEMENTS

We are thankful to the developers of Bentley Inc for their support and constructive discussions. This research was performed at the Geo-Database Management Centre at GIST, OTB, Delft University of Technology, The Netherlands.

REFERENCES

- ARENS, C., J.E. STOTER, AND P.J.M. VAN OOSTEROM, 2005, Modelling 3D spatial objects in a geo-DBMS using a 3D primitive. *Computers & Geosciences*, 2, pp. 165-177
- AUTODESK, 2005, ObjectARX Developer's Reference. Available online at www.autodesk.com (accessed 02.01.2006)
- BENTLEY, 2005, JMDL Developer's Guide, Available online at www.bentley.com/support (accessed 02.01.2006)
- BREUNING, M. AND S. ZLATANOVA, 2006, 3D Geo-DBMS. In *Large-scale 3D Data Integration: Challenges and Opportunities*, Zlatanova and Prosperi (Eds.), pp. 88-113 (Boca Raton: Taylor&Francis, 2006)
- OGC, 2001, OpenGIS Consortium, The OpenGIS Abstract Specification, Topic 1: Feature Geometry (ISO 19107 Spatial Schema), OpenGIS Project Document Number 01-101, Wayland, Mass., USA.
- OOSTEROM, VAN P., J. STOTER, W. QUAK AND S. ZLATANOVA, 2002, The balance between geometry and topology. In *10th International Symposium on Spatial Data Handling*, D.Richardson and P.van Oosterom (Eds.), pp. 209-224 (Berlin: Springer-Verlag, 2002)
- OOSTEROM, VAN P., J. STOTER, AND E. JANSEN, 2006. Bridging the worlds of CAD and GIS. In *Large-scale 3D Data Integration: Challenges and Opportunities*, Zlatanova and Prosperi (Eds.), pp. 9-36, (Boca Raton: Taylor&Francis, 2006)
- ORACLE, 2003a, Oracle JDBC developer's guide 10g release 1. Available on-line at <http://www.oracle.com/technology/documentation>, (accessed 02.01.2006)
- ORACLE, 2003b, Oracle ODBC developer's guide 10g release 1. Available on-line at <http://www.oracle.com/technology/documentation>, (accessed 02.01.2006)
- PENNINGA, F., 2005, 3D Topographic Data Modelling: Why Rigidity Is Preferable to Pragmatism, in 'Spatial Information Theory', Cosit'05', Vol. 3693 of Lecture Notes on Computer Science, Springer, pp. 409-425
- PIEGL, L. AND TILLER W., 1997, The NURBS Book 2nd Edition, Springer-Verlag
- PLÜMER, L., 2004, Bridging the gap between GIS and CAAD — geometry, referencing, representations, standards and semantic modelling, *GIM International*, 12-15, July 2004.
- SCHUTZBERG, A., 1995 Bringing GIS to CAD — A Developer's Challenge, *GIS World*, 8 (5), pp. 48-54
- SHEPHERD, I.D.H., 1990, Mapping with desktop CAD: a critical review, *Computer-Aided Design*, Vol. 22(3), pp. 136-150
- PU, S. 2005, Managing Freeform Curves and Surfaces in a Spatial DBMS, MSc Thesis, TU Delft, 2005, 77 p. Available at <http://www.gdmc.nl/publications> (accessed 02.06.2006)
- ZLATANOVA, S. AND J. STOTER, 2006, The role of DBMS in the new generation GIS architecture. In *Frontiers of Geographic Information Technology*, S.Rana and J. Sharma (Eds.), pp. 155-180 (Berlin: Springer-Verlag, 2006)