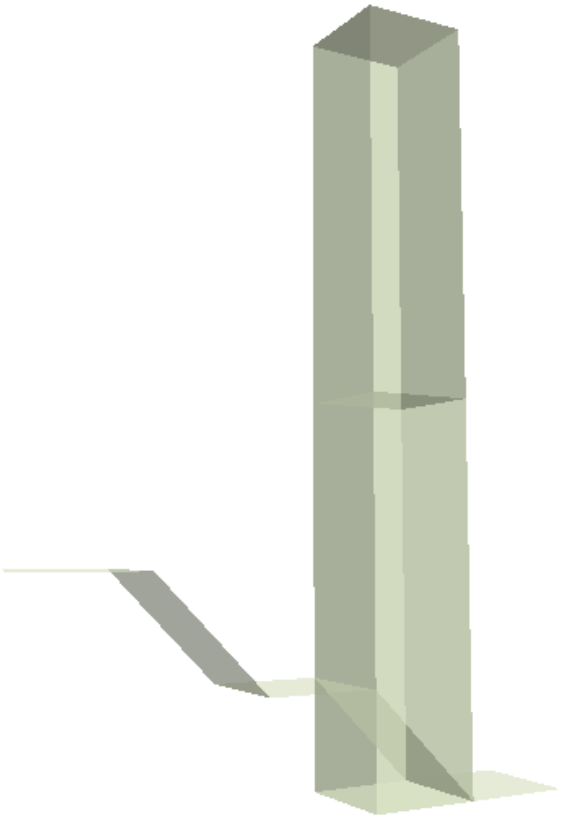
Centre for Geo-Information

Thesis Report GIRS-2003-33

Taking GIS Into Another Dimension

Developing A 3D Model, The “Glueface” Approach



August 2003

Richard Flierman

WAGENINGEN UNIVERSITY

WAGENINGEN UR

Taking GIS into another dimension

Developing a 3D geo data model, The “Glue-Face” Approach

Richard Flierman

Thesis

For achieving the academic degree

Master of Science in Geo-Information Science

Wageningen University

The Netherlands

Supervisors:

Dr. Ir. Ron van Lammeren
John Stuiver

Wageningen, The Netherlands, August 2003

Abstract

The world as we see it is 3 dimensional. Users of GIS systems want to model the earth as realistic as possible. Currently, most of the GIS systems are only capable of modelling the earth in 2D and 2½D, and so the demand for a 3D GIS is rapidly increasing.
Therefore, the objective of this thesis is to create a 3D integration model that is able to integrate 2D, 2½D and 3D data. To achieve this objective three steps are defined: 1) defining the problems for designing a 3D integration model, 2) designing a 3D integration model and 3) implementing and applicating the designed model.

A 3D GIS must perform at least the same functions as a 2D GIS. The main difference is that in a 3D GIS the data has to be modelled in a 3D way. The main advantage of 3D GIS over 2D GIS is the increase of realism. Currently, the 3D part in a GIS is based on bringing different data from different databases together on a computer screen. So the current situation is almost purely based on 3D visualisation.

The main difficulty in integrating 2D, 2½D and 3D data is the difference in topology. When going from a 2D world to a 3D world, the mutual relationships between objects are increasing enormously. This topological problem has to be overcome with the model to be designed.

The best modelling technique for 3D modelling is Object Oriented (OO). Because it was planned to use Microsoft Access as Database Management System (DBMS) the best alternative Object Relational modelling.

To overcome the topological difficulties in integrating the different data types, the Glue-Face model is designed. The Glue-Face model is called this way because it makes use of so called “gluefaces”. A glueface is a face where two objects are “glued” together in a topological way. The Glue-Face model is a modest integration approach and doesn’t describe not full 3D topology.

After implementing the Glue-Face model, the model is tested. This is done by means of querying the model with simple selection queries.

From the results of testing the model can be concluded that the objective of this thesis is achieved: A 3D integration model has been designed, which can deal with the topological difficulties.

Acknowledgements

In front of you lies my thesis report “Taking GIS Into Another Dimension”. I wrote this thesis report for achieving the academic degree Master of Science (MSc.) in Geo-Information Science at Wageningen University.

Many people contributed to the realization of this thesis report. I am grateful to all those people who made some time free to help me and supporting me in achieving my goal.

Especially my sincere gratitude goes to my supervisor Dr. Ir. Ron van Lammeren, for his advice and support and to Ir. John Stuiver for his technical advise, both from Wageningen University, Department of Geo-Information and Remote Sensing.

Finally, I want to thank my friends, family and my girlfriend, Heleen Boxebeld, for their support and patience during this thesis.

Wageningen, August 2003

Richard Flierman

Contents

List of Figures

List of Tables

1 Introduction

The world as we see it at a certain moment is 3-dimensional (3D). This means that the world and all the objects on it basically consist of three dimensions, namely length, width and height. In mathematical terms these three dimensions are called the x, y and z-axis. The x and y axes usually represent length and width and the z value is standing for the height (or depth) (fig. 1.1).

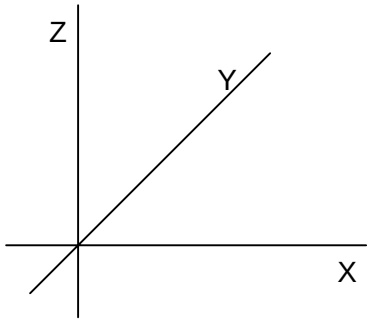


Figure 1.1: Georeferencing by X-, Y- and Z-values

So every object on earth should at least be represented in x, y and z values. These values are better known as the *Geometry* of an object. Between the different objects on earth are relationships. For example, a tree stands on a parcel, one house is standing behind another house, a car is standing in front of the church, etc. These relationships are better known as the *Topology* of objects.
For years, scientists are trying to model the very extensive and difficult geometry and topology of the real world for the use in computer programmes. Such computer programmes are often revered to as *Geographic Information Systems (GIS)* and are, amongst other things, used to analyse and visualize *Geodata* (real world modelled data). Modelling the real world is not an easy task and trying to design and to constantly improve these models to meet the desires of the GIS users.

1.1 Problem definition

At the moment, mainly 2-dimensional (2D) GIS systems are used. Such 2D systems describe the real world in 2 dimensions, length and width (x and y values). In such a system the real world is so to say “beaten flat” and a lot of the real world geometry and topology is lost.
Nowadays, most of the 2D GIS systems offer the possibility to use geodata in 2½-dimensions (2½D). This data can be used to model relief and can be seen as floating 2D geodata, like a “flying carpet” (fig 1.2).

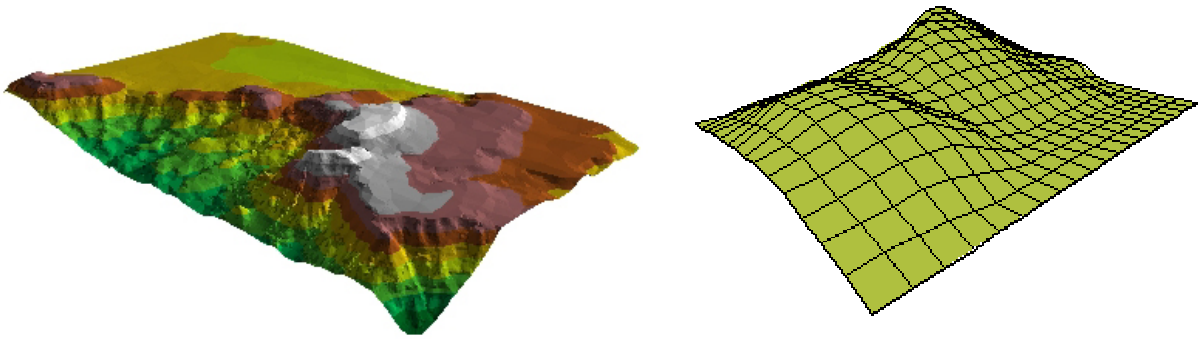


Figure 1.2: Two examples of a 2½D geodata-set.

The 2D GIS systems have extensive 2D and 2½ geometry and topology and are well capable of analysing and visualizing these geodata. The latest developments offer the users of 2D GIS systems the possibility to visualize their geodata in 3D by simply adding a z value to their geodata. It suits a better visualization of geodata but doesn't offer any 3D editing and analysing capabilities.
Recently, for a more realistic analysis and visualization, the demand is growing for a 3D GIS system. Such a system has to have a much more extensive geometry and topology. Unfortunately such a system in a GIS environment is rarely available.
To overcome this problem, 3D geodata is spread between several different systems. For example, one system is used for the storage of the 3D geodata (a database system) and one system is used to visualize the data. This separation over several systems is laborious and is costing extra time and effort.
The challenge is to integrate the 2D, 2½D and 3D geodata into one data model. Such a data model must make the editing, analysis and visualization of an integrated use of 2D, 2½D and 3D geodata possible.

1.2 Objective of the research

The general objective of the research is to integrate geo data sets that describe, in a 2, 2½ and 3 dimensional way, real world objects. For the thesis, the general objective is tapered on four main groups of questions (A, B, C and D).

A) Theoretical questions based on literature research

- Why 3D GIS?
- What is the current state of 3D GIS?
- What trends and development?
- Why is integration difficult?
- Which data structures are there?

B) Questions related to the conceptual system design via conceptual design of a data integration model by using "ER" or "UML" and a viewer.

- What are the principles of modelling?
- Which data models are already developed?
- Which conditions are there?

C) Questions related to the system implementation by making use of Microsoft Access and a viewer.

D) Questions related to the use of the application by simple queries.

1.3 Reader instructions

Although there are seven chapters, this thesis report can be divided into basically four parts, based on the research objectives. The first and largest part deals with the theoretical questions (research objective A). This is described in the chapters 1, 2 and 3. Research objective B, the design of a conceptual data model, is the second part of the thesis. This part is presented by the chapters 4 and 5. The principles of modelling is described in the Appendix. The implementation and application of the designed model, research objective C and D, is the last part and is dealt with in chapter 6. Finally, chapter 7 comes with a conclusion and recommendations.

Chapter 1 gives an introduction of the thesis with the problem definition and the research objective.

Chapter 2 explains the definition of a 3D GIS and why and by whom 3D GIS is needed and what the advantages of 3D GIS are compared to 2D GIS. The current situation of 3D GIS is also expressed in this chapter.

Chapter 3 clarifies the problem for data integration by comparing the differences in topology, geometry and data-structure between 2D, 2½D and 3D data.

Chapter 4 presents a few existing data models for integration and introduces a new solution: The “ Glue-Face” model.

Chapter 5 explains the implementation and application of the “Glue-Face” model with the use of Microsoft Access, Oracle Spatial 9i and ArcScene.

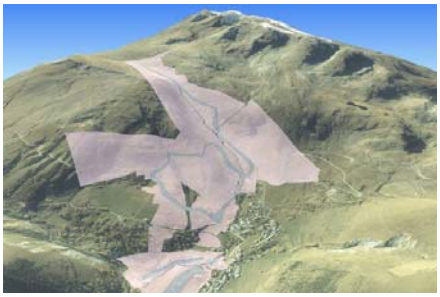
Chapter 6 presents the conclusion of the thesis and comes with recommendations and suggestions for further research.

2 3D GIS

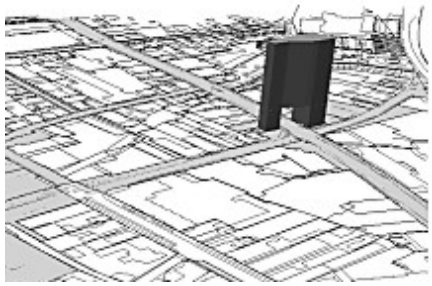
The demand for modelling real world phenomena in 3D is growing every day. Areas in which 3D GIS can be important are [ZLAt1], [ZLAT2], [ZLAT3], [PIL], [KOF], 3D urban planning, civil engineering, environmental monitoring and engineering, telecommunications, public rescue operations, landscape planning (landscape architecture), geological and mining activities, transportation monitoring, real-estate market, hydrographical activities, utility management, military applications, archeology and meteorology. Figure 2.1 shows some examples of 3D GIS application.

Applications of 3D GIS for these areas are for example [ZLAT1], [KOF]:

- Analysis on the effect of new communication and transportation networks, factories, etc., in terms of pollution, noise etc.
- Real estate offices use 3D data for the maintenance of ownership of apartments and offices located in one building, extended means for visual exploration.
- The planning of sites of cultural and architectural importance, where a 3D visualisation is an essential part of the entire design.
- Governmental/municipal authorities require tools to perform administrative tasks (including traffic planning, disaster preparedness etc.) more efficiently.
- 3D models can help to improve citizens participation in a decision making process.
- Telecommunication companies need 3D data to calculate wave propagation in urban environments.
- Architects want photo-realistic models of existing buildings to plan new ones and to visualize the result.
- Vendors of traffic navigation systems need 3D data to improve the accuracy and ease of use of their systems.
- Tourism agencies want to offer virtual reality models of destinations they are offering.



A



B

Figure 2.1: Examples of 3D GIS applications. A and B show an example of a 3D cadastre and C and D show an example of 3D construction plan

C

D

2.1 What is 3D GIS?

What is 3D GIS? This question will be answered by first explaining the question “What is (2D) GIS”. This will be done with the use of some definitions of the term GIS as described in several GIS (internet) glossary’s.

Definitions of (2D) GIS:

- Geographic information system: An organized collection of computer hardware, software, geographic data, and personnel designed to efficiently capture, store, update, manipulate, analyse, and display all forms of geographically referenced information [GLOS1].

- Geographic information system: A computer system for capturing, storing, checking, integrating, manipulating, analysing and displaying data related to positions on the Earth's surface [GLOS2].

- An acronym for 'Geographic Information System' which describes a computer system capable of assembling, storing, manipulating, and displaying geographically referenced information, i.e. data identified according to their locations [GLOS3].

- Geographical Information System: A Geographical Information System is a computer-based system consisting of hardware, software, data and applications. The system serves to capture and update spatial data, to store and maintain, to analyse and present these data in an alphanumeric and graphic form [GLOS4].

As can be concluded from the definitions above; the tasks of a (2D) GIS are mainly (see also [ZLAT1], [ZLAT2], [OOS1]):

1) Data capture (input)
2) Data storage and database management
3) Data manipulation and analysis
4) Data presentation (output)

These tasks can be described as follows [GIS glossary's], [ZLAT1], [ZLAT2] [OOS1]:

1) **Data Capture** is the acquisition and input of spatial data to the system. This is the collection of data in both digital and analogue form and its transformation into an appropriate standardised format for entry into the GIS database. Data sources include paper maps, satellite images, aerial photographs, field notes and other paper records. These need to be converted into a digital form, if they are not already, before the GIS can make use of them. Procedures such as digitising, scanning and manual keyboarding are involved in this conversion process.

2) **Data Storage and database management** Data is stored and indexed within a database system, which facilitates shared access to the data, and which maintains the reliability, security and integrity of the data by controlling access to it and supervising updates. Database management thus tends to be at the heart of a GIS ensuring controlled and coordinated data retrieval and analysis. Ideally the data set should be structured in such a way as to be independent of the applications, which access it.

3) **Data manipulation and analysis** With data manipulation, obviously, the manipulation of the GIS data is meant. The three main classes in data manipulation are querying, transforming and processing. Data analysis is the process of identifying a question or issue to be addressed, modeling the issue, investigating model results, interpreting the results, and possibly making a recommendation. Some analysis functions are: Terrain analysis, geometric computations, overlay, buffering, zoning and sorting. The process of modeling, examining, and interpreting model results. Spatial analysis is useful for evaluating suitability and capability, for estimating and predicting, and for interpreting and understanding. There are four traditional types of spatial analysis: topological overlay and contiguity analysis, surface analysis, linear analysis, and raster analysis.

4) **Data Presentation** is the presentation of all the generated information or results such as in the form of maps, graphs, tables, reports, etc. Output from a GIS can be in the form of an interactive computer display of the graphic and/or attribute data or as traditional paper maps, tables, graphs and reports. Automated cartography techniques are employed to produce the graphical output from GIS. Commonly used output devices include pen plotters, ink-jet printers, electrostatic plotters, laser printers and screen copiers. Export to other systems such as statistical packages, word processors and multi-media will usually require the GIS to be able to convert internal data into other standard formats.

Now that we know the functions of a 2D GIS, follows here a definition of a 3D GIS:

A 3D GIS system should be able to model, represent, manage, manipulate, analyse and support decisions based upon information associated with three-dimensional phenomena [ZLAT2].

From the above-mentioned information and definitions, we can conclude that a 3D GIS must perform at least the same functions as a 2D GIS. The difference is that 3D GIS has to deal with three dimensional phenomena/data, so the data has to be modelled in a 3D way.

2.2 Advantages of 3D GIS over 2D GIS

In this section the advantages of 3D GIS over 2D GIS will be discussed. Therefore, a few quotes of other researchers will be used. As we look at the difference between 2D GIS and 3D GIS we immediately think at of the (extra) third dimension. Because the real world is actually 3D, it is obvious that modelling the real world in a 3D GIS becomes more realistic then modelling it in a 2D GIS.
"This not only makes the user interface easier to use and reduces training times for professional GIS users, it also makes GIS data accessible to new groups of users with no previous GIS experience (e.g. in public information systems). Thus, expensive data can be used more widely. 3D data is also necessary for a number of new applications, which have not been possible using 2D data (e.g. demographic analysis simulation which help in disaster prevention, emergency training and delivery etc.)" [KOF]
Thus in a 3D GIS the analysis and visualisation of the real world, for example for environmental science, is much more realistic then with a 2D GIS. Another advantages of 3D GIS over 2D GIS is that the programmes are easier to work with. This is because we understand better what we see. This advantage is also devoted to the increased realism.
"Van Driel (1989) recognized that the advantage of 3D lies in the way we see the information. It is estimated that 50 percent of the brain's neurons are involved in vision. What's more, it is believed that 3D displays stimulate more neurons: involving a larger portion of the brain in the problem solving process. With 2D contour maps, for example, the mind must first build a conceptual model of the relief before any analysis can be made. Considering the cartographic complexity of some terrain, this can be an arduous task for even the most dextrous mind. 3D display, however, simulates spatial reality, thus allowing the viewer to more quickly recognize and understand changes in elevation" [SWA]
As can be concluded from this section, the advantage of 3D GIS over 2D GIS lies particularly in the increased realism. Other advantages are merely based on this increased realism.

2.3 Why is it difficult to realise 3D GIS?

The difficulties in realising a 3D GIS can roughly be divided in designing a conceptual model, spatial analysis, visualisation and the collection of data. First of all the conceptual model. The common understanding is that the conceptual model (and its corresponding logical model) is the key element in a 3D GIS [ZLAT1]. The conceptual model provides the methods for describing real-world objects and spatial relationships between them [ZLAT1]. The conceptual model must be able to integrate different data types into one database. Because of the "extra" (third) dimension the geometry and especially the topology becomes much more complex. Therefore, designing a good conceptual model also becomes more complex. The more complex topology is also the cause of another difficulty, namely the spatial analysis. Most of the spatial analysis performed in a GIS is based on spatial relationships. One can imagine that if the topology becomes more complex, the analysis based on this topology will also become more difficult. The visualisation of 3D data and the navigation through it, can also form a difficulty. Because of the large 3D scenes with an extended geometry, especially when textures are used, quick visualisation and navigation through the scene can become very slow. This difficulty is merely a computer performance problem. The computer systems are constantly becoming faster and faster, thus to overcome this problem is just a matter of time. The last difficulty that will be mentioned, the collection of (3D) data, is particularly a financial problem. Modelling in 3D drastically increases the cost of data acquisition, as compared with 2D [ZLAT1]. This is mainly caused by the extended geometry. Although automatic detection and 3D reconstruction is getting better, it still does not work in acceptable quality [PIL]. So most of the acquisition has to be done manually, which will cost extra time and therefore money.

2.4 Current situation

The 3D part of a GIS is currently based on bringing different data types from different databases together on a computer screen. So the current situation is so to speak “integration by means of visualisation’ (fig 2.2). The different data is not linked together and 3D analysis is barely or not possible.

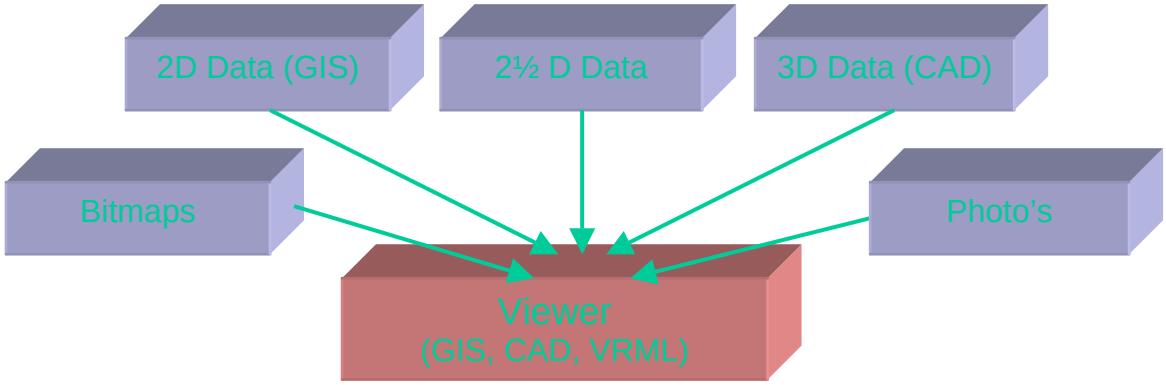


Figure 2.2: Integration by means of Visualisation

The situation is very obscure. There are for example 2D maps, textures (bitmaps), photo’s, 3D features like buildings and trees etc. brought together on one computer screen. As a picture the result is nice but to work with all the different data is complicated.
The current situation is also expressed in figure 2.3. This figure shows that (geo-) data belonging to different (geo-) data-models is actually used (brought together) in a visualisation model.

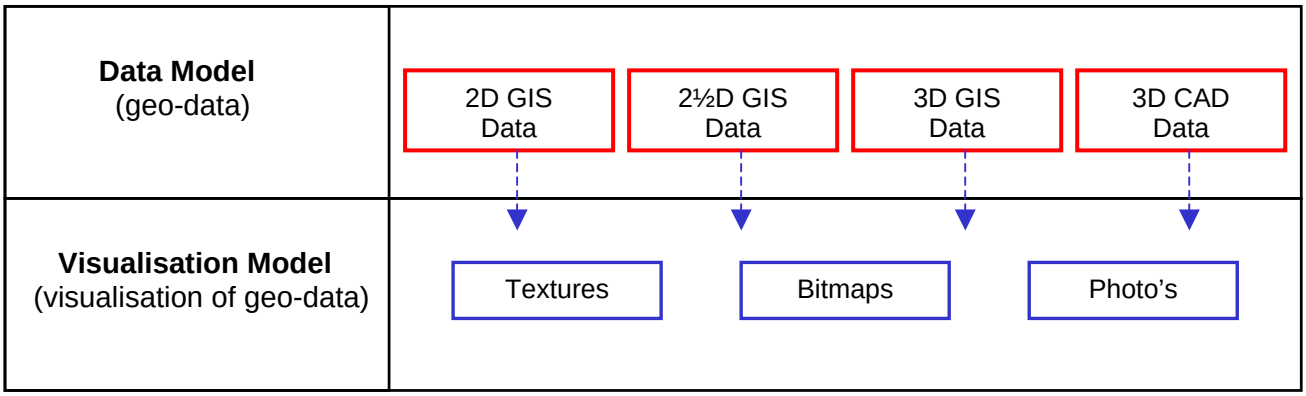


Figure 2.3: Present data use for integration

3 Data Structures

In this chapter the differences in describing real world phenomena in digital format (geo-objects) will be shown. There will be looked at the differences in how 2D, 2½D and 3D are described and what possible problems these differences can give for the integration of these three data types.

For this research, the following hypothesis is formulated, also based on section 2.3 of the previous chapter:

"The main problem in integrating 2D, 2½D and 3D data is the difference in topology. Because the topology is different, the geometry, data model and georeference will also be different".

This hypothesis is based on the following reasoning: The real world as we see it has a very extensive topology. One can think of things like *behind* that building, *inside* that house, *on* the roof, *under* the bridge, *along* the road, and so on. If we try to describe the real world in 2D (figure 3.1A) a lot of topology we have in the real world can't be represented. Imagine you beat everything you see in the real world to a flat plane. Then, topology like *under* and *behind* is not possible anymore. 2½D (figure 3.1B) data is nothing more then "floating" 2D data, like a flying carpet. So like 2D data, 2½D data also has a limited topology. When the real world is described in 3D (figure 3.1C), the topology extends. In a 3D world, topology like *inside*, *under* and *above* are becoming possible again.

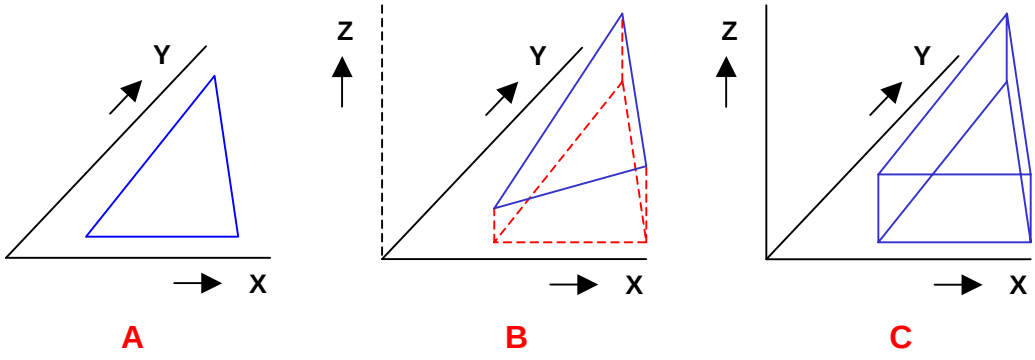


Figure 3.1: An example of real world phenomena described as respectively a 2D (A), 2½D (B) and 3D (C) object

The hypothesis will be checked by comparing the data structure of some datasets on primarily *topology*, and *geometry*. This will be done in the next section.

3.1 2D Data

First of all we will look at 2D data. The geographic entities in a 2D GIS are based on two different types of data: spatial data and thematic data (fig. 3.2) [OOS1]. Spatial data has two components: geometric and topological data.

As explained in chapter one, geometric data is used to represent the spatial component (coordinates) of geographic objects and topological data is used to describe the relationships between these geographic objects.
Thematic data gives extra information about a spatial object. Like the name of a river or the land use of a parcel

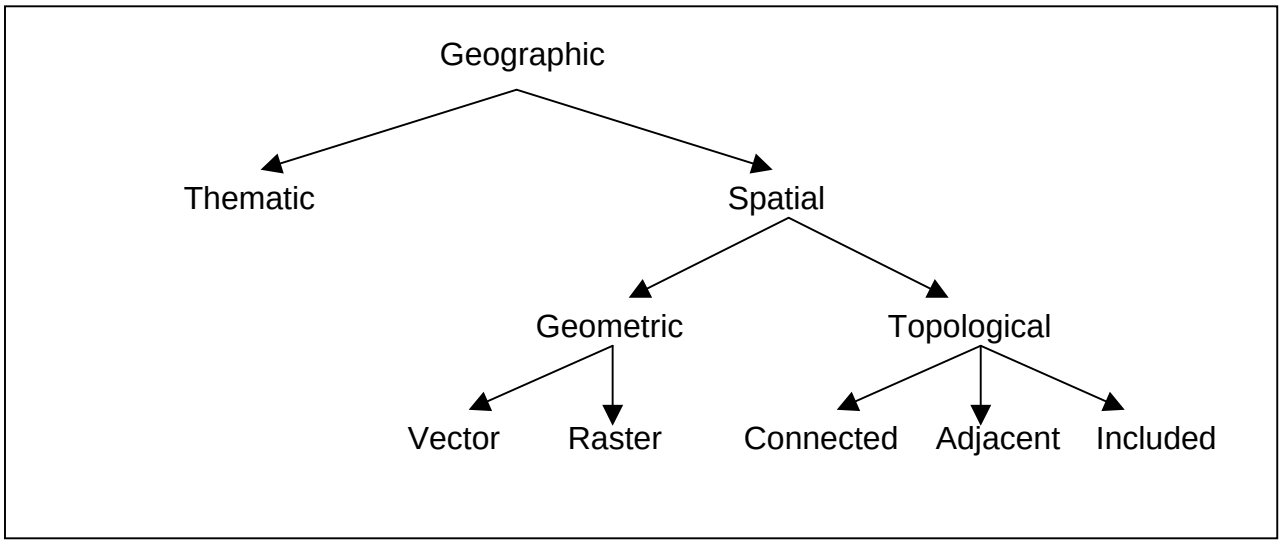


Figure 3.2: The Hierarchy of Geographic Data Types (OOS1).

As can be seen in figure 3.2, 2D geometric data can basically be divided in 2D vector data and 2D raster data. 2D vector data can be subdivided into point (node), polyline (line, arc) and polygon (area).
The three fundamental topological components are connectivity (connected), containment (included) and adjacency. Examples of queries concerning topological relationships are: which areas are neighbours of each other (adjacency), which polylines are connected and form a network of roads (connectivity) and which lakes lie in a certain country (containment) [OOS1]. The figures 3.3 and 3.4 are showing examples of respective 2D vector and 2D raster geometry and topology.

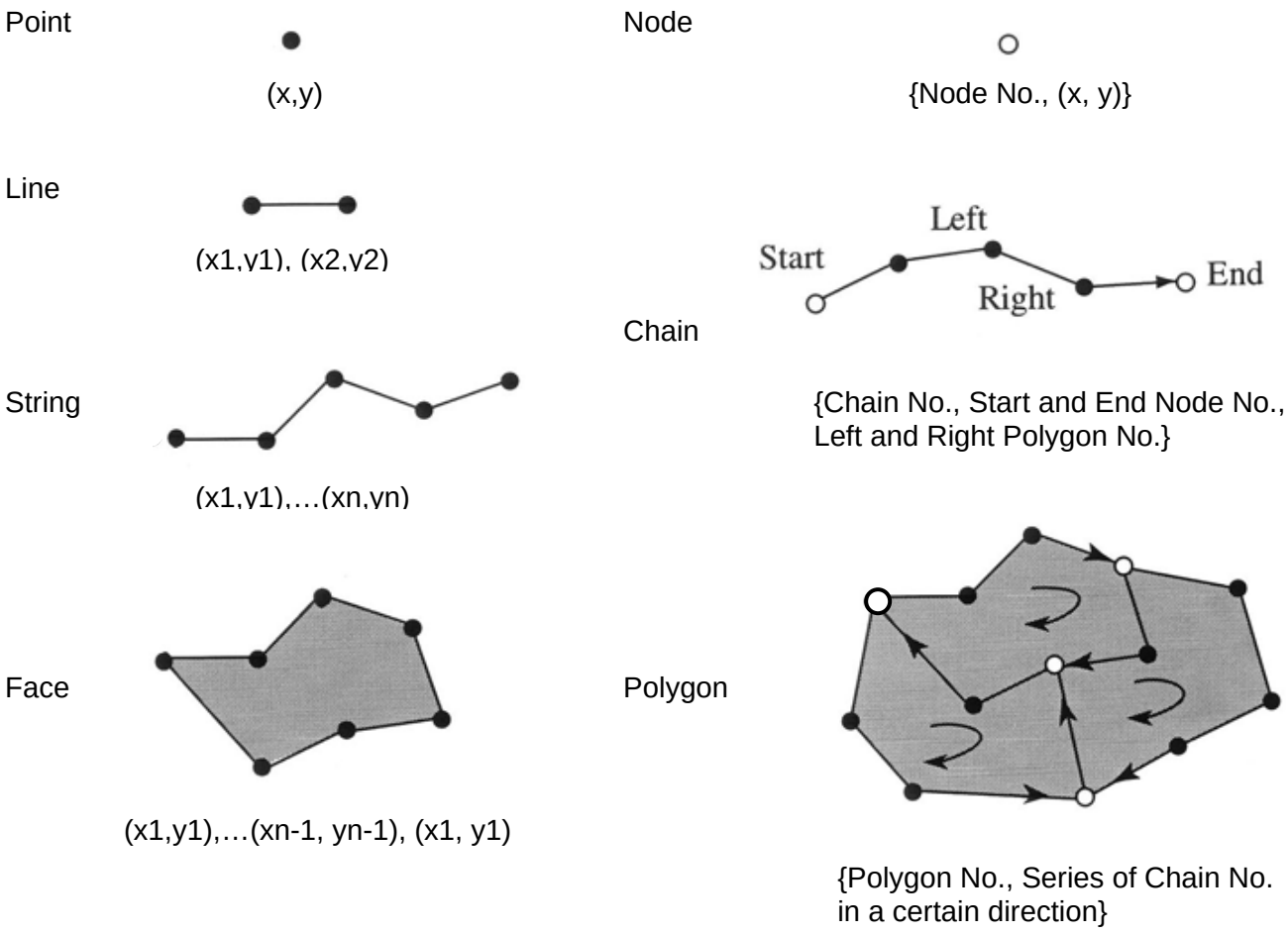


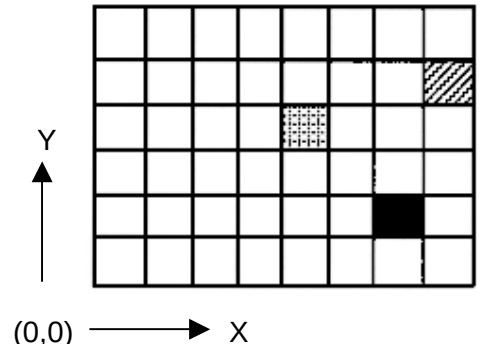
Figure 3.3: The Geometry and Topology of 2D Vector Data [MUR]

In figure 3.3 the three geometric objects, point, line and area (polygon) are showed. The geometry of a point is given by two dimensional coordinates (x, y), while line (and string, a sequence of lines) and area are given by a series of point coordinates [MUR] The 2D topology is described by node, chain and polygon. A node is an intersect of more than two lines or strings, or start and end point of a string with a node number. A chain is a line or string with chain number, start and end node number and left and right polygons. Finally, a polygon is an area with polygon number and series of chains that form the area in a certain order.

In order to analyse a network consisting of nodes and chains, the following topology is Built [MUR]:
Chain: chain ID, Start Node ID, End Node ID, Attributes
Node: Node ID, (x, y), adjacent chain ID's (positive for to node, negative for from node)

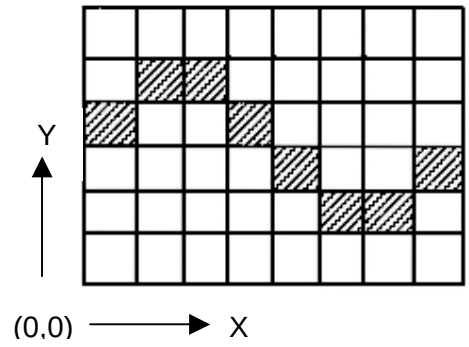
In order to analyse not only a network but also relationships between polygons, the following topology and geometry is built [MUR]:
Chain geometry: Chain ID, Start Coordinates, Point Coordinates, and End Coordinates
Polygon topology: Polygon ID, Series of Chain ID, in a certain order
Chain topology: Chain ID, Start Node ID, End Node ID, Left Polygon ID, Right Polygon ID

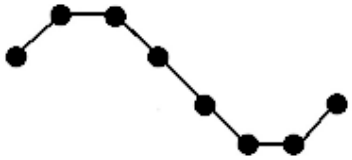
Point Objects



(x, y) = (5,3); (7,5); (8,2)

Line Objects

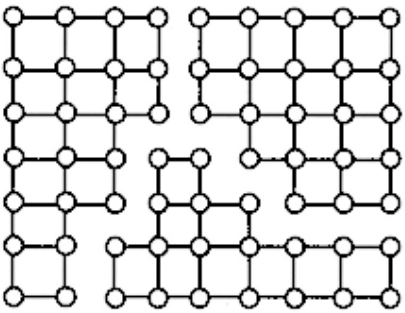




(1,3), (2,2), (3,2), (4,3)
(5,4), (6,5), (7,5), (8,4)

Area Objects

A	A	A	A	B	B	B	B
A	A	A	A	B	B	B	B
A	A	A	C	C	B	B	B
A	A	A	C	C	C	B	B
A	A	C	C	C	C	C	C
A	A	C	C	C	C	C	C



(4A, 4B), (4A, 4B), (3A, 2C, 3B),
(3A, 3C, 2B), (2A, 6C), (2A, 6C)

Geometry **Topology**

Figure 3.4: The Geometry and Topology of 2D Raster Data [MUR]

Figure 3.4 shows the geometry and topology of 2d raster data. The basic geometric object in raster data is a pixel. All objects are built out of pixels. All the pixels in the same data set are of the same size. The geometry of raster data is given by point, line and area objects [MUR]. A point is given by a single (x,y) tuple a point ID and attributes. A line is given by a series of coordinate tuples, and like a point, a line has a line ID and attributes. Finally, an area is given by a group of coordinates and, like point and line, by an area ID and attributes. Area objects are typically given by “run length” that rearranges the raster into the sequence of length (or number of pixels) of each class [MUR].

As mentioned before the pixel (or grid cell) is the primary spatial entity within a grid. Each cell is square, has the same size as other cells in the grid and contains a numeric value representing the spatial variable at that location. The coordinate system of a grid is the same of that for other geographic data. The rows and columns are parallel to the x- and y-axes of the coordinate system. Since each cell within a grid has the same dimension as other cells, the location and area covered by any cell is easily determined by its row and column. The coordinate system of a grid is thus defined by cell size, the number of rows and columns, and the x, y coordinates of the lower-left corner (the origin).

3.2 2½D Data

For describing the topology and geometry of 2½D data, the TIN will be used. A TIN is an acronym for triangulated irregular network. A TIN is used for representing continuous (terrain) surfaces, like height models. The main components of a TIN (fig. 3.5) are triangles, nodes and edges. Nodes are locations defined by x, y and z values from which a TIN is constructed. Triangles are formed by connecting each node with its neighbours. Edges are the sides of triangles. The TIN data structure is defined by two elements: a set of input points with x, y, and z values, and a series of edges connecting these points to form triangles. Each input point becomes the node of a triangle in the TIN structure, and the output is a continuous faceted surface of triangles. The triangles are constructed according to a mathematical technique. The most used technique is the Delaunay triangulation. The technique guarantees that a circle drawn through the three nodes of any triangle will contain no other input point and will be the smallest circle that could be drawn between 3 adjacent points.

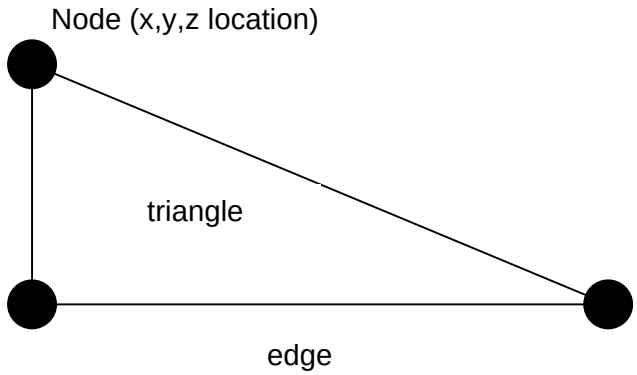
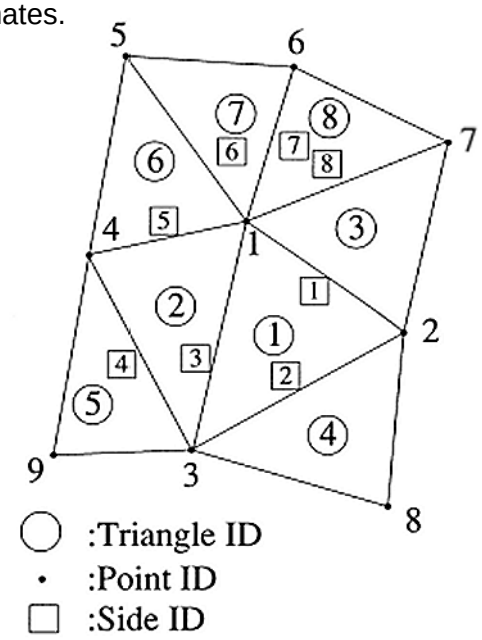


Figure 3.5: The structure of a TIN triangle

The topology of TIN's (2½D data) is similar to 2D vector data (fig. 3.6). The topology can be described in three different ways triangle based, point (node) based or side (edge) based. The triangle-based structure is defined by a triangle ID, the three triangle coordinates (clockwise direction) and the neighbour triangles. This is more or less similar to the topology of (2D) polygons. The point-based structure is defined by a point ID, it's coordinates and the neighbour (connected) points. Finally, the edge-based structure is defined by a side ID, The start and end point of the edge, and the neighbour triangles. This is more or less the similar to the (2D) chain topology. So merely the difference between the topology and geometry of 2D vector data and 2½D TIN data is that the TIN nodes are defined by x, y, z coordinates instead of only x and y coordinates.



(a) Triangle based structure

ID	Point Coordinates	Neighbors
①	(x_1, y_1, z_1), (x_2, y_2, z_2), (x_3, y_3, z_3)	②, ③, ④
②	(x_1, y_1, z_1), (x_3, y_3, z_3), (x_4, y_4, z_4)	①, ⑤, ⑥
⋮	⋮ ⋮ ⋮	⋮ ⋮ ⋮

(b) Point based structure

ID	COORD	Neighbors
1	(x_1, y_1, z_1)	2, 3, 4, 5, 6, 7
2	(x_2, y_2, z_2)	1, 7, 8, 3
⋮		

(c) Side based structure

Point : ID, (x, y, z)				
Trianglis : ID, (P1, P2, P3)				
Side	ID	(P1, P2)	$T_r l$,	$T_r r$
⋮	[1]	(1, 2)	③,	①
	[2]	(2, 3)	④,	①
	⋮	⋮	⋮	⋮

Figure 3.6: The TIN topology

3.3 3D Data

Like in 2D data, 3D geometric data can be divided into vector data and raster data. 3D vector data can be subdivided into node (point), edge (line), face and volume. There are also two supporting entity types: rings and shells. The 3D entities will be explained below.

Node

A 3D node (fig.3.7) is like a 2D or 2½D node, a zero-dimensional spatial entity that defines a location in 2D or 3D space and defined by a single coordinate tuple (xyz) [TRO]
2D topology recognizes two types of nodes: connected nodes and entity nodes.
A connected node is found at each endpoint of an edge, and is topologically linked to all edges that it bounds [TRO]. An entity node is contained within the interior of a face, and is not located on or topologically linked to any edge that bounds that face [TRO].
3D topology defines a third node type, the space node. A space node is contained within the interior of volume, and is not located within or topologically linked to any face which bounds that volume [TRO]. Because of the volumes in 3D topology, nodes can be classified in different ways. For example, a node can be the endpoint of an edge (connected node) and also be situated within a volume (space node).

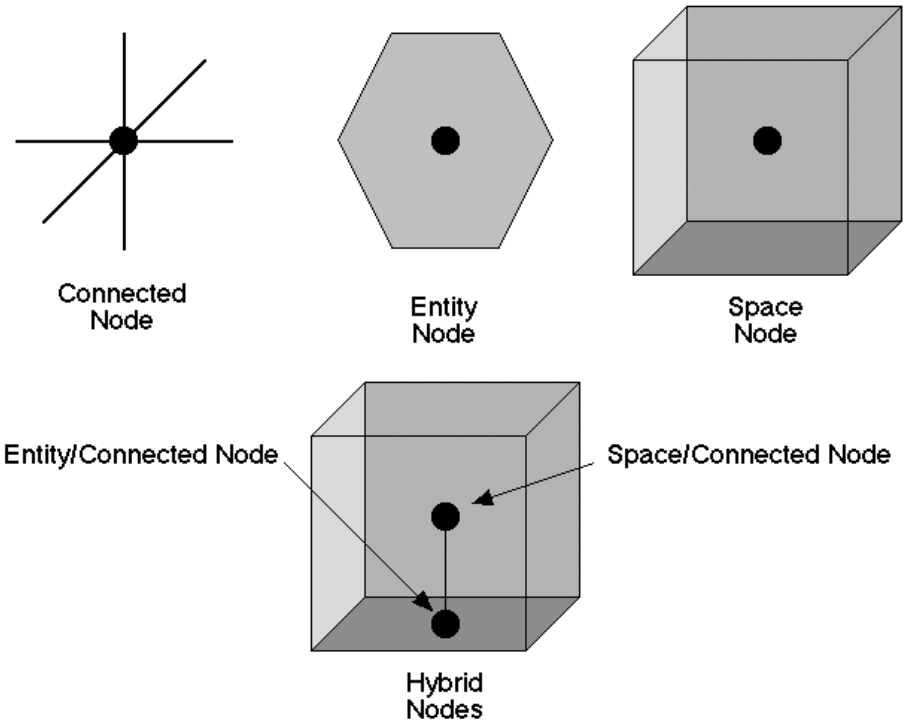


Figure 3.7: 3D node entity [TRO]

Edge

An edge (fig. 3.8) is a one-dimensional spatial entity that defines a path through 2D or 3D space and is defined by an ordered collection of two or more distinct coordinate tuples (xyz) [TRO].

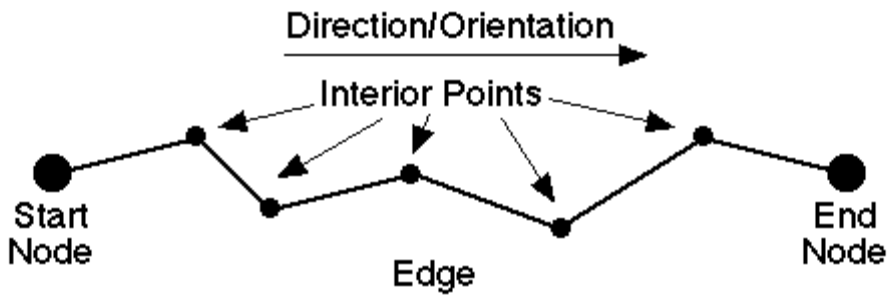


Figure 3.8: 3D edge entity [TRO]

An edge is bounded by a node at each of its two endpoints and may not intersect with or overlap itself or any other edge [TRO].

Face

A face (fig. 3.9) is a two-dimensional spatial entity that defines a closed area in 2D or 3D space and is defined by [TRO]:

- a collection of one or more edges that bound the face
- a collection of zero or more nodes that are contained within the face,
- a collection of zero or more interior points that are analogous to the interior points of an edge.

Faces may contain "holes", each of which is defined by an inner boundary consisting of an ordered collection of edges [TRO].

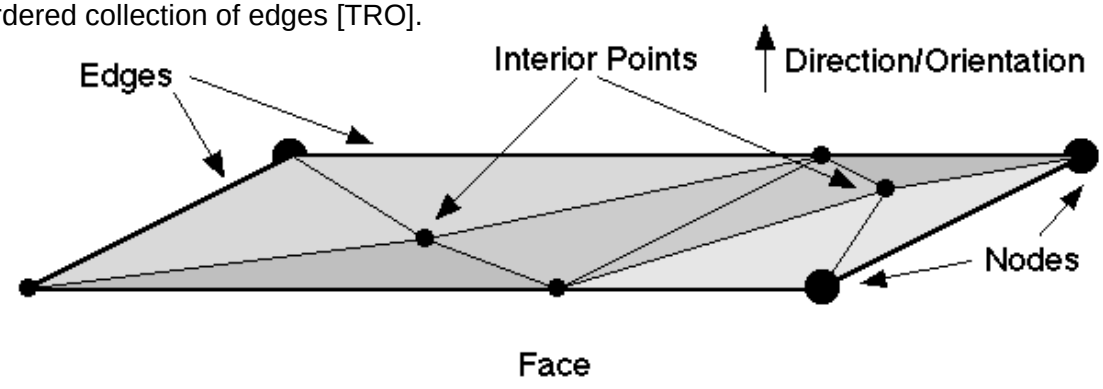


Figure 3.9: 3D face entity [TRO]

Ring

A ring (fig. 3.10) is a sequentially connected set of edges that bound a face [TRO]. Mostly a ring is forms a face and can be divided in outer ring and inner ring. An inner ring is a ring within a face (face 3 in fig. 3.10). Then again face three has an outer ring formed by edge 4 and 5. So the edges of face three form the outer ring of face three and one of the inner rings of face 2. So for one face it is an inner ring, for the other it is an outer ring. Any edges that are contained in the face, but which are connected to the outer boundary of the face, are included in the outer ring (edge three in fig. 3.10) [TRO]. An inner ring mostly forms/contains a face but it is possible that an inner ring does not contain a face. In that way, an inner ring can represent a hole (fig. 3.10). A collection of one or more edges which are connected to one another, and which are contained within the face, but which are not connected to the outer boundary of the face, form an inner ring even if they do not enclose an area (edge 9 in fig. 3.10) [TRO].

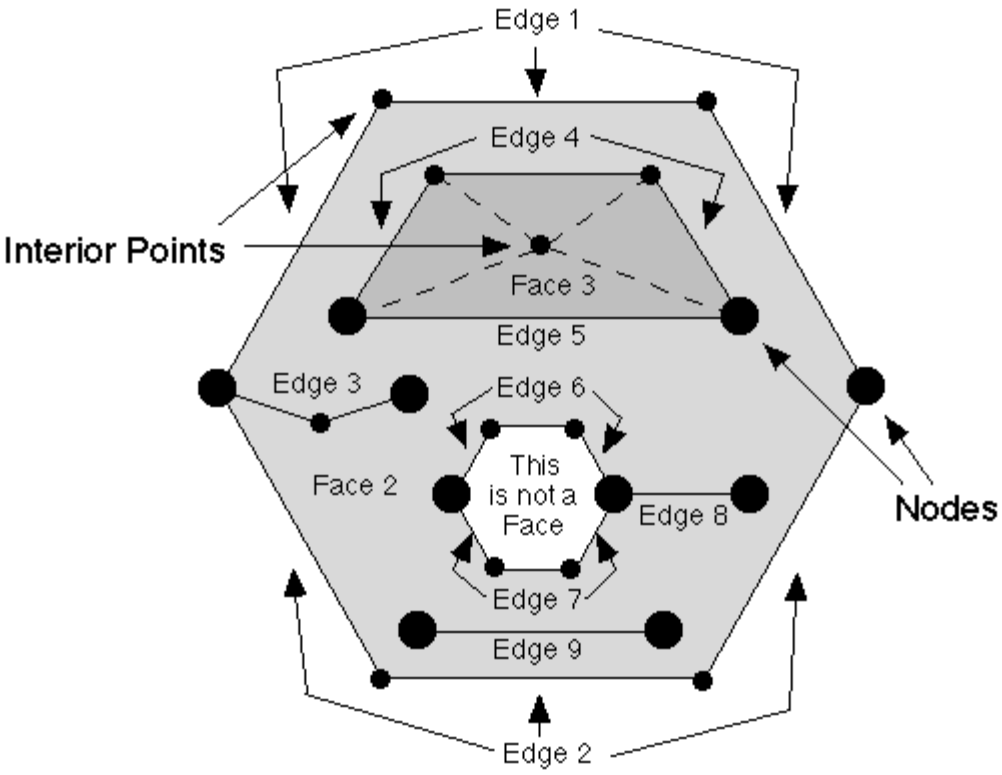


Figure 3.10: 3D ring entity [TRO]

Volumes

A volume (fig. 3.11) is a three-dimensional geometric primitive that is composed of the closed region of space bounded by a collection of faces and may be topologically linked to nodes, edges, and faces [TRO].

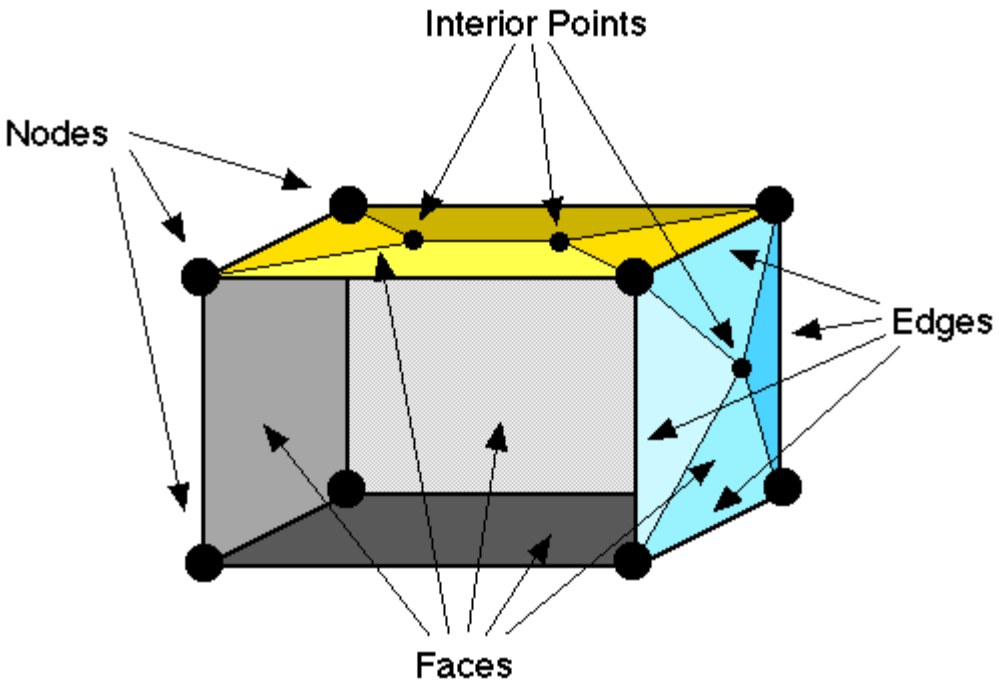


Figure 3.11: 3D volume entity [TRO]

Shells

A shell (fig.3.12) is an unordered collection of two or more faces that bound a volume [TRO]. Each shell is associated with a specific volume and a face that forms the boundary between two volumes appears in exactly two shells, one for each of the volumes that it bounds [TRO]. Like the ring, a shell can also be an inner shell or an outer shell. The outer shell of volume two is the inner shell of volume one (see fig. 3.12). The inner shell of volume 1 forms a "bubble" within this volume.
In a certain defined space every volume within that space can be seen as an inner shell of that space.

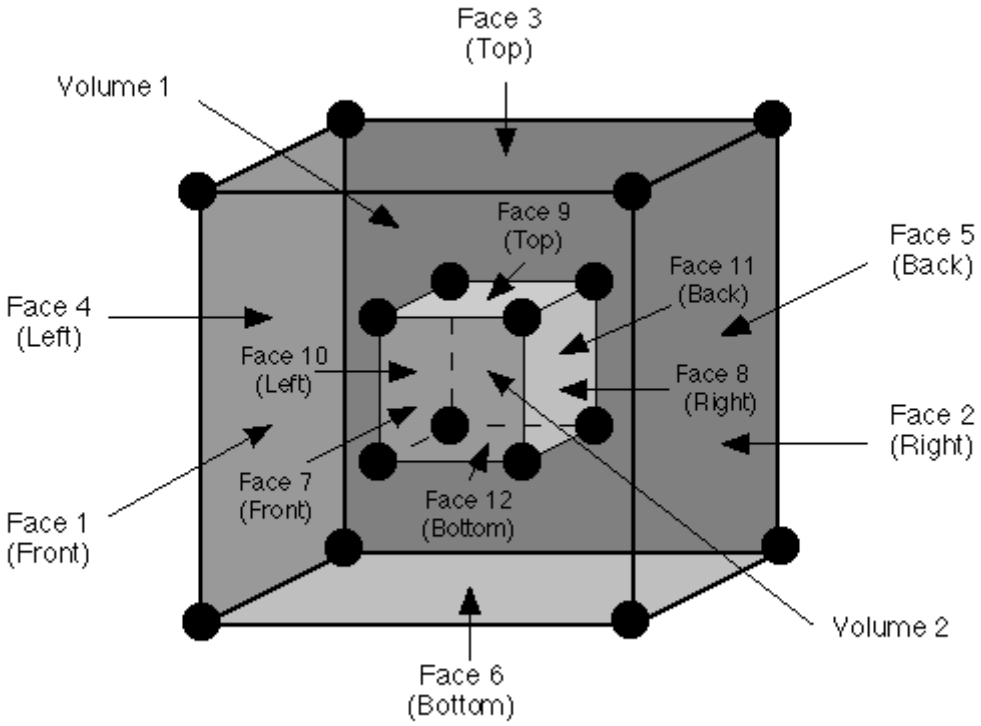


Figure 3.12: 3D shell entity [TRO]

3.4 3D topological relationships

Node-edge relationships

As shown in figure 3.13, an edge has two nodes, an end and a start node. The start and end node divine the direction. A node can be connected to one or more edges.
In 2D topology, these connected edges can be ordered in a counter clockwise (or clockwise) direction around the node, starting with an arbitrary edge [TRO]. In 3D topology this is not so easy. An edge can be connected to a node from any direction, so the edges are hard to order. When volumes are defined, subsets of the connected edges that bound a given volume can be identified, and the connected edges in each of these subsets can be ordered, just as in the 2D case [TRO].

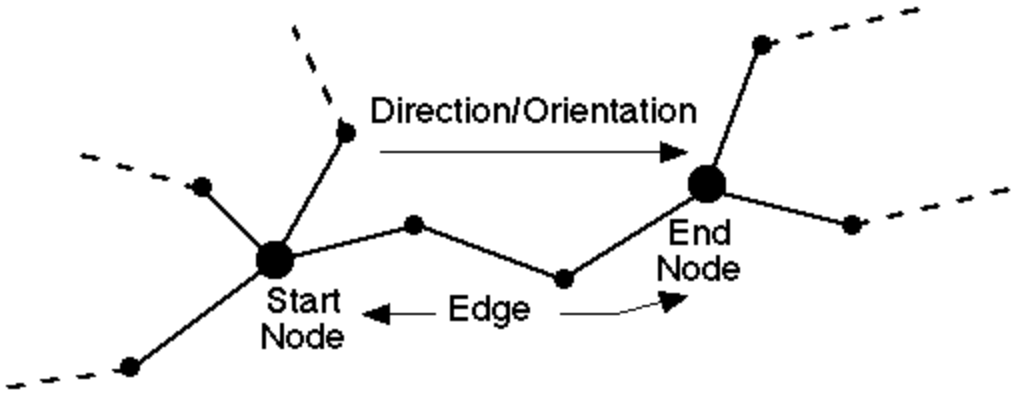


Figure 3.13: 3D node-edge relationship [TRO]

Node-face relationships

A face can be defined by the nodes it contains (contained nodes, and nodes can be defines by the face containing that node (containing face) (fig. 3.14).

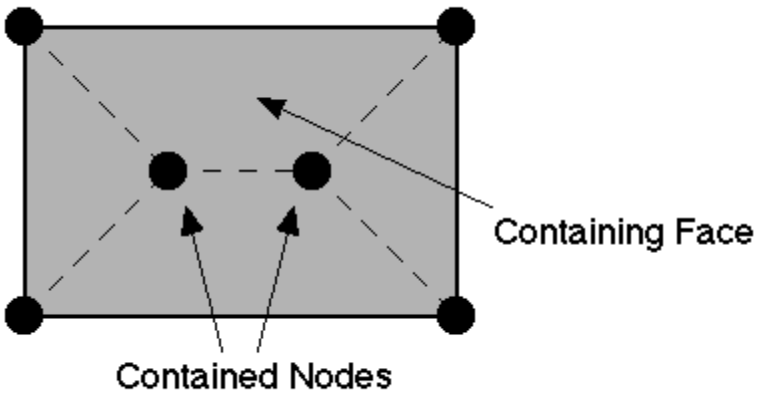


Figure 3.14: 3D node-face relationship [TRO]

Node-volume relationships

Like the node-face relationships, a volume can be defined by the nodes it contains (contained nodes, and nodes can be defines by the volume containing that node (containing volume) (fig. 3.15).

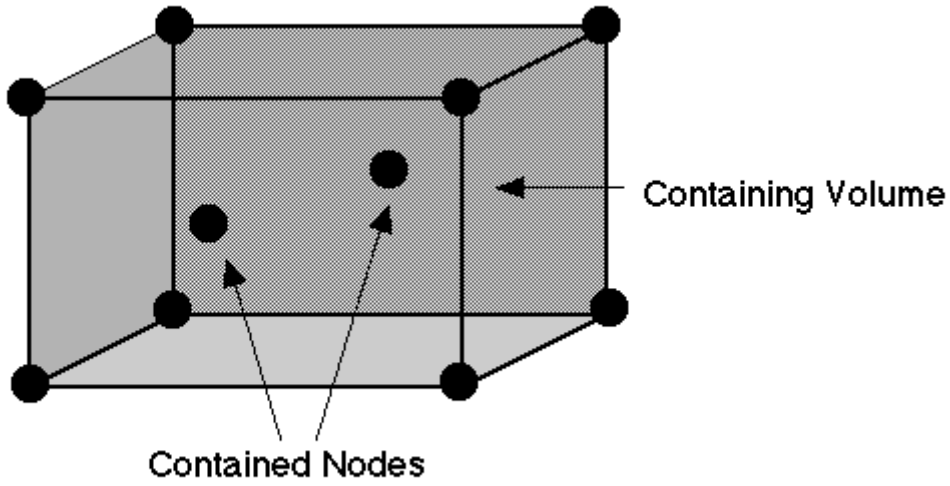


Figure 3.15: 3D node-volume topology [TRO]

Edge-face relationships

In 2D topology, every edge is contained by two faces (fig. 3.16). These two faces are called the left- and right face, depending on the direction of the edge. In 3D topology, there can be more then two faces sharing the same edge (fig. 3.16).
These bordered faces are ordered in a counter clockwise direction around the edge, starting with an arbitrary face, relative to the direction of the edge [TRO]. The many-to-many relationship between edges and faces is the primary source of the 3D spatial data model's additional complexity [TRO].

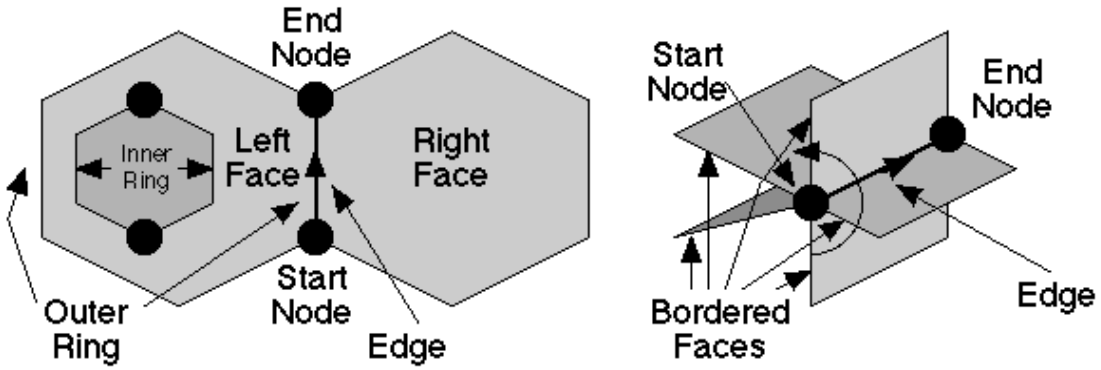


Figure 3.16: 3D edge face relationship [TRO]

Edge-volume relationships

Under full 3D topology, each edge that bounds one or more faces is also associated with an ordered collection of one or more volumes, as shown in figure 3.17 [TRO]. These bordered volumes can be ordered in a counter clockwise direction around the edge, alternating with the bordered faces, starting with an arbitrary volume, relative to the direction of the edge, just as edges are ordered around a connected node in 2D [TRO].

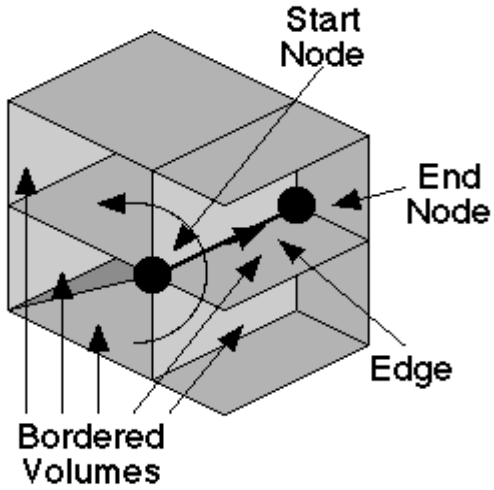


Figure 3.17: 3D edge-volume relationship [TRO]

Volumes can also contain edges that not bound any faces. These edges are so called “floating” edges. This volume can be the defined space (universe volume).
A volume is defined by one or more collections of edges. A collection of edges is associated with each of the shells that bound a volume, forming the outer rings of the faces that make up that shell (fig. 3.12) [TRO].

Face-volume relationships

Under full 3D topology, each face is associated with exactly two volumes, which are called its top volume and bottom volume and are defined relative to the orientation of the face (fig. 3.18) [TRO]. Each volume is associated with one or more collections of faces, each of which composes a shell [TRO]. For all volumes, except the universe volume, there is exactly one outer shell, which must contain two or more faces [TRO]. A volume may have zero or more inner shells, each of which also must contain two or more faces [TRO].
Because the collection of faces, which makes up a shell cannot, in general, be ordered such that each face is connected to the next by a common edge, this one-to-many relationship must be explicitly defined [TRO].

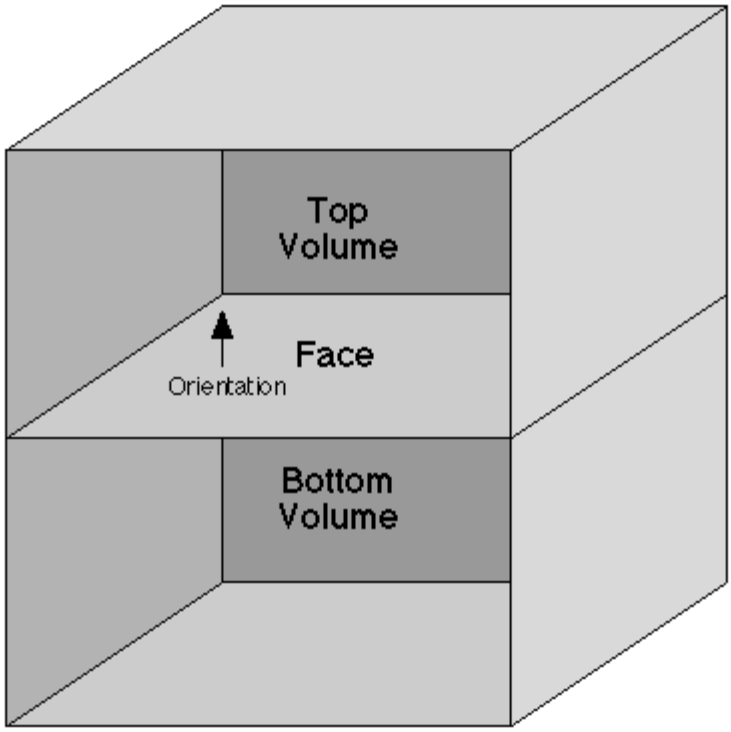


Figure 3.18: 3D face-volume relationship [TRO]

Floating/dangling nodes & edges

Full 2D topology allows for edges, or collections of connected edges, that are "dangling" within a face (i.e., attached to a boundary of the face at only one end) (fig. 3.19) [TRO]. A dangling edge belongs to the ring to which it is attached. A collection of dangling edges can enclose (form) a face. The collection of edges that form that face is the inner ring of the face that contains them and the outer ring of the face they enclose. A floating edge or a collection of connected floating edges within a face is also considered as an inner ring within the face containing them.

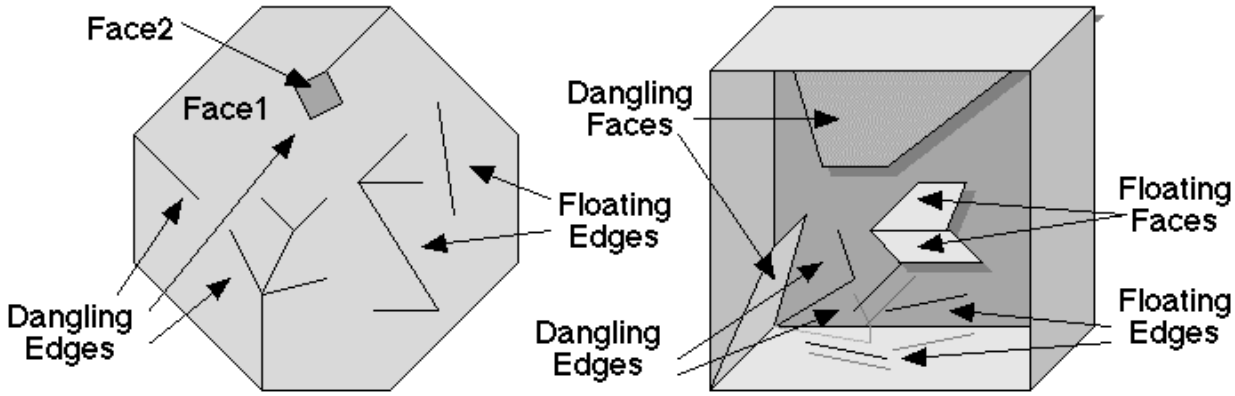


Figure 3.19: Floating/dangling nodes & edges [TRO]

Similar to 2D topology which allows dangling and floating edges within a face, 3D topology allows for "dangling" or "floating" faces within a volume [TRO].
Like the 2D topology, a dangling face or a collection of connected dangling faces is attached to either the outer shell or an inner shell of that volume along one or more (but not all) of it's bounding edges and is considered to be a part of the shell to which it is attached. [TRO].
A floating face is like a floating edge not attached to any other shell of the volume along any of its bounding edges. The floating face is also considered as an inner shell of the volume that contains the floating face. Floating edges within a volume are also allowed in 3D topology. Similar to 2D topology, an edge is considered a floating edge when it is contained within a volume and does not bound any of the faces which form the outer shell, or any inner shell, of that volume [TRO].

3D Raster Topology

3D raster data uses, similar to 2D raster data one basic entity to describe topology. 2D raster data uses pixels and 3D raster data uses voxels (fig. 3.20). Therefore the topology and geometry of 2D and 3D raster data are very much similar.

Figure 3.20: Basic raster entities

3.5 Conclusion

As can be concluded from this chapter, the topology of vector data is much more complicated then of raster data. The topology of 2D vector data and 2½D TIN data is similar. As showed in table 3.1, the topology of 3D vector data compared to 2D/2½D vector data is much more extensive. The difference in geometry between 2D, 2½ and 3D data is not very big. So coming back to the hypothesis at the beginning of this chapter that, *"The main problem in integrating 2D, 2½D and 3D data is the difference in topology"* ,one can conclude that for integrating 2D and 2½D (TIN) data, the topology is not that big of a problem. Integrating 2D, 2½D and 3D data and storing them into one database will give much more problems and will not be directly possible because of the topological difference. So therefore can be concluded that the hypothesis is correct.

2D/2½D entities	**3D entities**
• connected nodes • entity nodes • edges • faces	• connected nodes • entity nodes • space nodes • edges • faces • volumes
2D/2½D relationships	**3D relationships**
• start & end nodes • connected edges • containing face • contained entity nodes • left & right faces • outer & inner rings	• start & end nodes • connected edges • containing face • containing volume • contained entity nodes • contained space nodes • contained entity edges • bordered faces • bordered volumes • outer & inner rings • outer & inner shells

Table 3.1: Comparison between the topology of 2D/2½D vector data and 3D vector data [TRO]

4 Model design

In this chapter the design of the conceptual- and logical model for the integration of 2D, 2½D and 3D data is described. Before we come to this, firstly some conceptual models of other researchers will be revealed. These particular models have been chosen because of their relevance with this thesis.

4.1 Conceptual models

The conceptual model is used to identify which objects are needed in the model with their characteristics and relationships. The conceptual model is independent of the database management system (DBMS) used for system implementation (Adam, 1997) (see Appendix).

First of all we will discuss the (2D) Formal Data Structure (FDS) Data Model proposed by Molenaar (1989) (fig. 4.1). The model has two entities, nodes and arcs. A line feature is described by an arc and a point by a node. An area feature is described by arcs and nodes. The FDS data model is designed for a single valued map, which means that “node” and “arc” can only appear in the description of one geometric object of the same dimension.

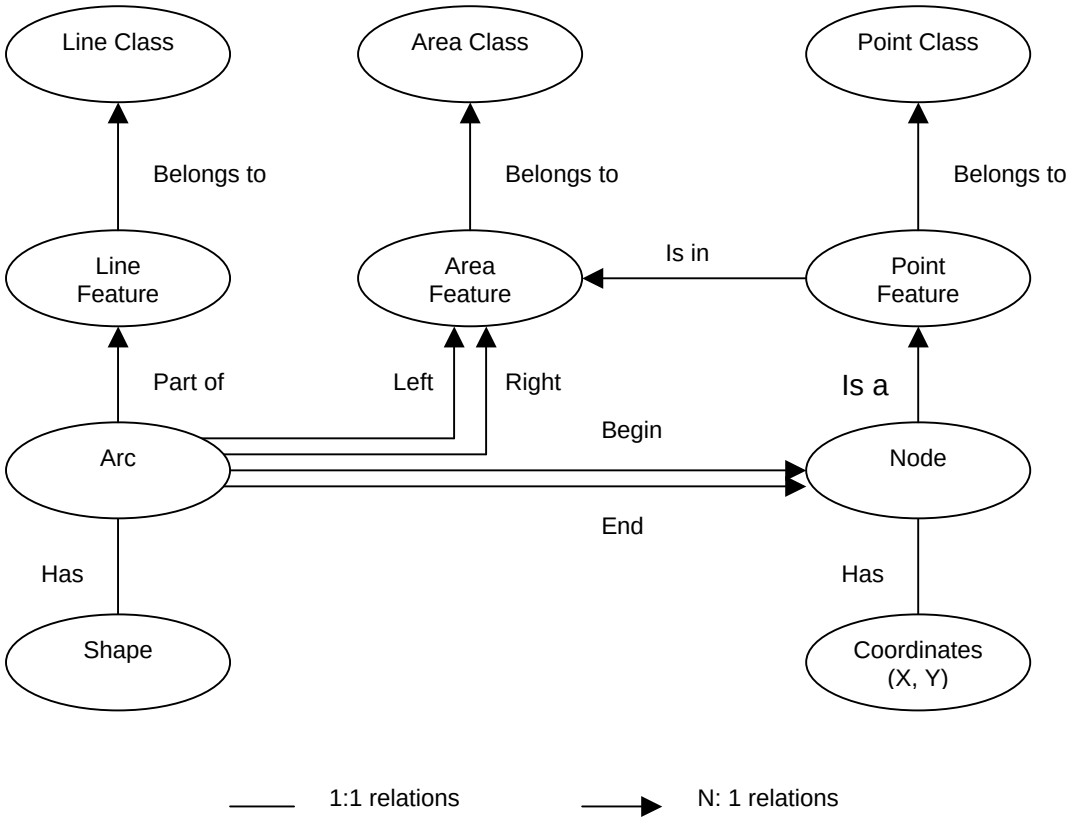


Figure 4.1: Formal Data Structure (FDS) Data Model (after Molenaar 1989)

The formal data structure geometrically abstracts spatial objects, such as monuments, roads, rivers, forest and parcels to points, lines and areas [PIL]. One of the most elegant aspects of the FDS is that it is possible to describe the topological relationships between the features in a formal manner, so the semantics of topological terms such as: neighbour, island, branching, crossing, intersecting, ending, inside, etc. are defined [OOS1]. Although the coordinates of every node can be 3D, the FDS model provides only 2D topology [PIL].

The second conceptual model which will be discussed is the 3D Formal Data Structure (FDS) Data Model proposed by Molenaar (1990) (fig. 4.2). This model is based upon the 2D FDS Data Model described in the previous part. This model can be used to model solid objects with 3D topology.
The 3D FDS Data Model permits the representation of spatial objects in the different dimensions 0D, 1D, 2D and 3D, that is the point, line, surface and body features respectively [PIL]. The model consists of three fundamental levels, feature (related to a thematic class), four elementary objects (point, line, surface and body) and four entities (node, arc, face and edge) [ZLAT1].

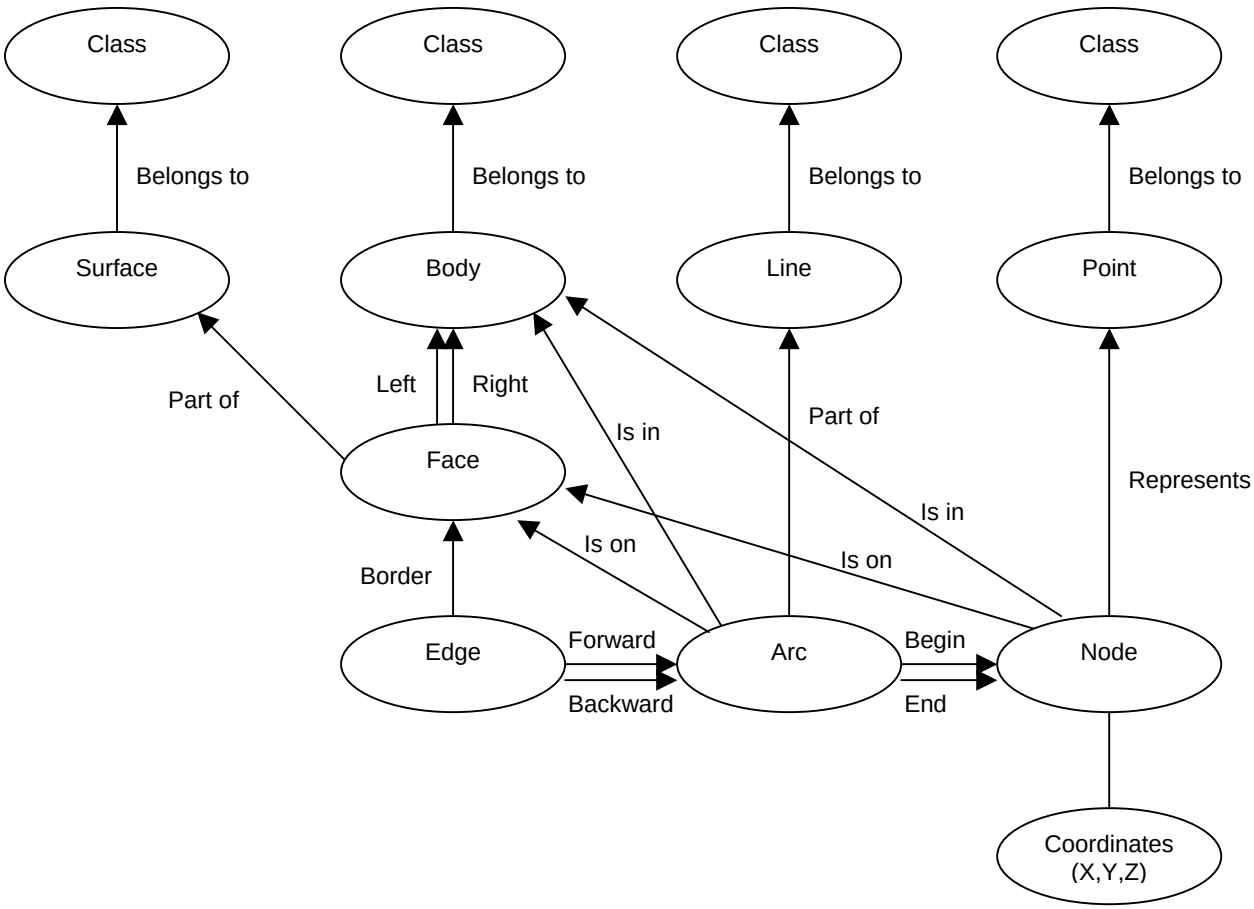


Figure 4.2: 3D Formal Data Structure (FDS) Data Model (after Molenaar 1990)

Edges are additional geometric primitives providing the link between arcs and faces and so permitting the unique reference to left and right bodies that are 3D features [PIL].
Like the 2D FDS, the 3D FDS is designed for a single valued map. So the primitives node, arc, face or edge, can only appear in the description of one geometric object of the same dimension [ZLAT1].

The last conceptual model which will be discussed is the TIN-based Data Model after Pilouk (1996) (fig. 4.3). This Data model is, like the 3D FDS, based upon the 2D FDS Data Model, and can be seen a an extension of this model. The TIN-based data model aims at integrating terrain relief (2½ D) with terrain features (2D).
Instead of the 2D FDS data model, an area feature is no longer linked to “arc” directly, but to its geometric primitive “triangle” which means that an area (surface) feature then consists of one or more triangles. In this model, each feature type maintains a feature-class (FC) link to exactly one class data type [PIL]. A geometric data type has a GF link (Geometry-Feature link) to a feature it composes. Between two related geometric entities are also geometry-geometry (GG) links [PIL1].

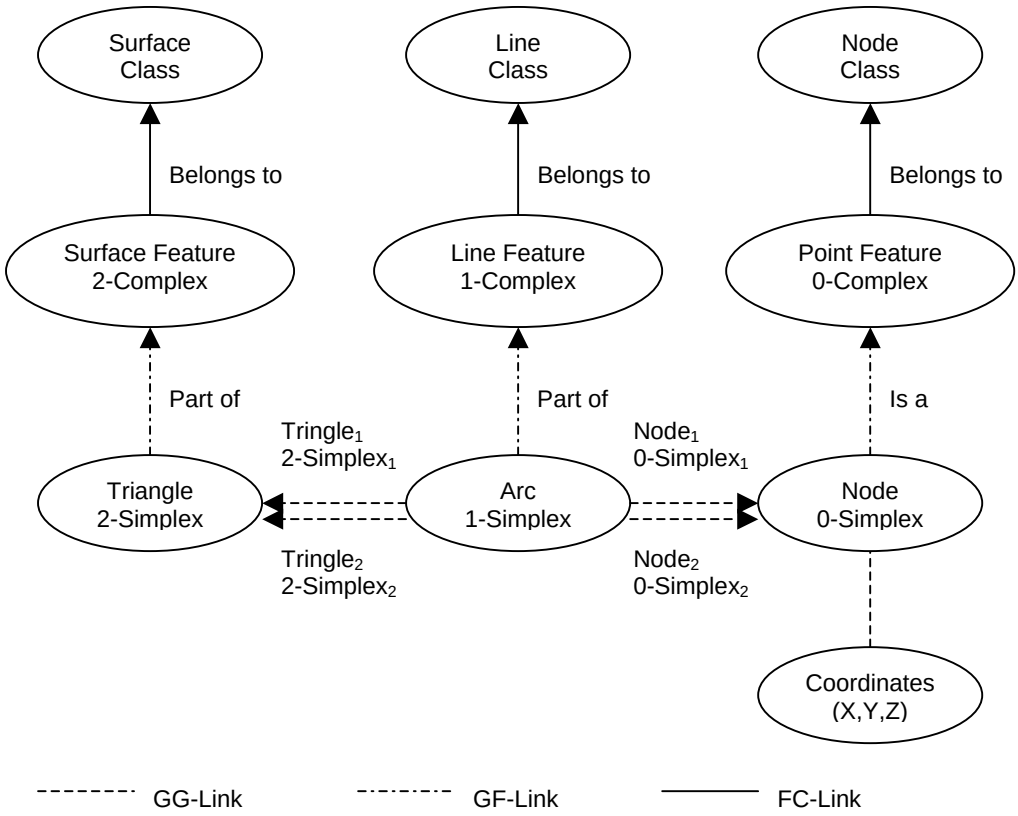


Figure 4.3: TIN-based Data Model (after Pilouk, 1996)

The TIN-based Data (integration) Model shows a 2½D solution and the 3D FDS Data Model shows a 3D solution. Because of the complexity of designing a 3D data model with full 3D topology, this thesis is focussing on a 3D integration model, for the integration of 2D, 2½D and 3D data.

As was concluded in chapter 3, to integrate the different data, the topology forms the most difficult problem. You have to know where an object is situated compared to another object. Therefore, designing the conceptual 3D integration model especially has been focussing on solving this topological problem. The conceptual model has to have solutions for several topological situations; it has to meet some conditions. These conditions are presented in the next section.

4.2 Conditions

With the creation of the model several topological conditions have to be taken into account. The model has to be able to integrate 2D, 2½D and 3D data, so many different situations can occur and have to be thought of. In the following part topological situations will be discussed, which, for this thesis, have been taken into account. The model to be designed has to have a solution for these topological conditions.

First of all, we can think of situations between 3D objects (fig. 4.4). For example, one 3D object can be on top of/under another 3D object, 3D object can be next to another 3D object, or a 3D object can be in front of/behind another 3D object. These 3D on 3D conditions mentioned here are the same conditions, just described from different viewing points.

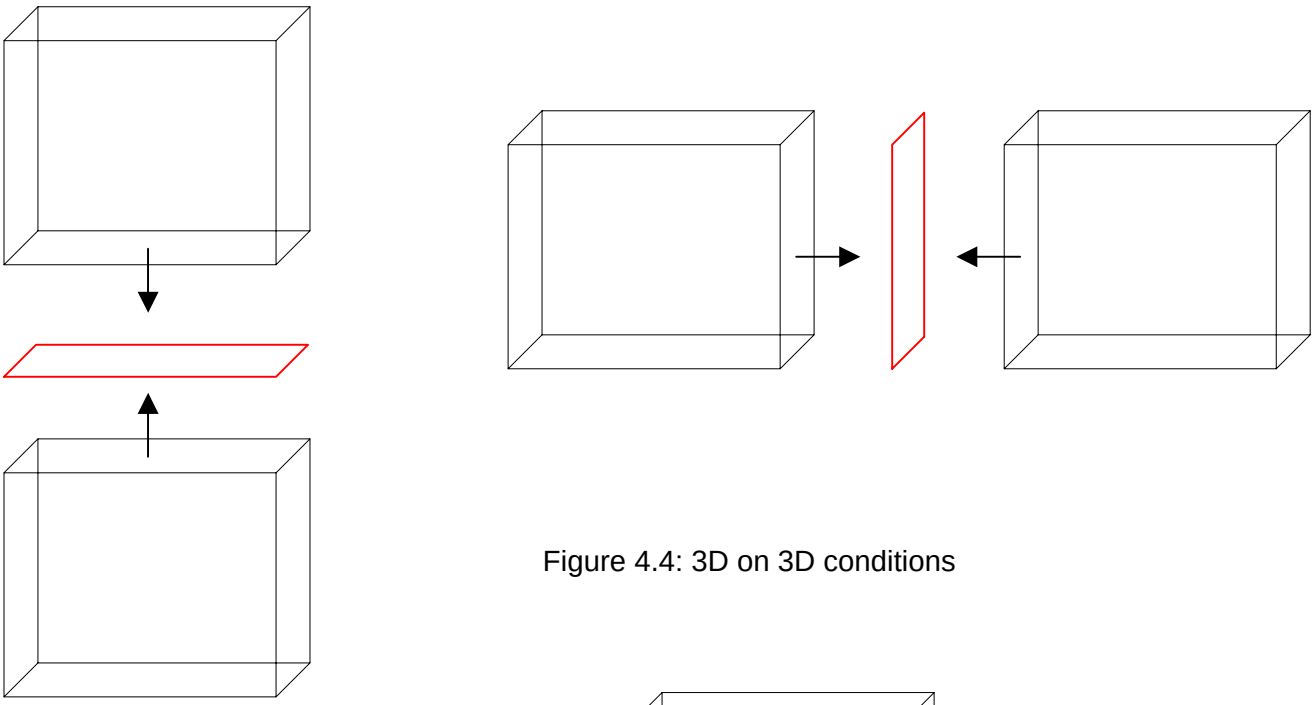
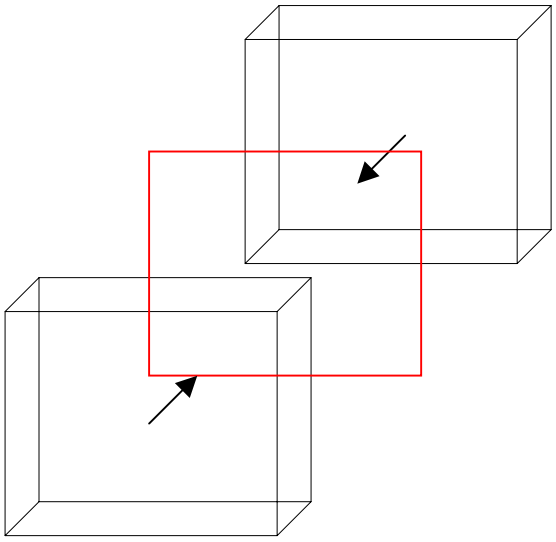


Figure 4.4: 3D on 3D conditions



As we can see in figure 4.5 and 4.6, we also have to deal with 3D objects which are situated on top of a 2D object or a 2½D object (TIN).

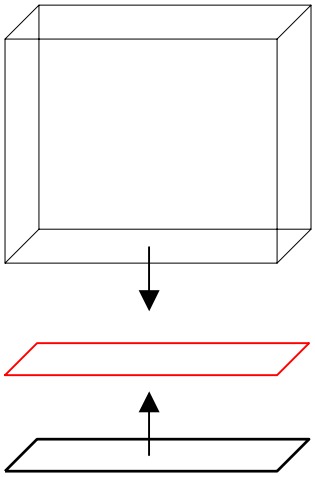


Figure 4.5: 3D on 2D condition

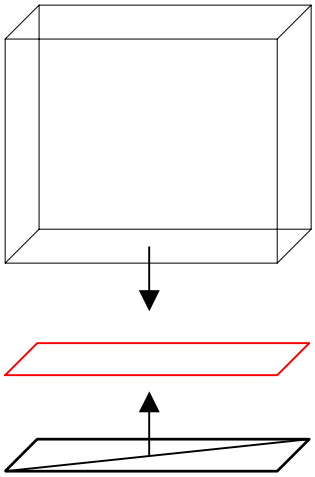


Figure 4.6: 3D on TIN condition

With the creation of the model, we also have to think of 2D objects that are situated on top of a 2½ D object (TIN) (fig.4.7) and 2D objects that are on top of/under another 2D object (fig. 4.8).

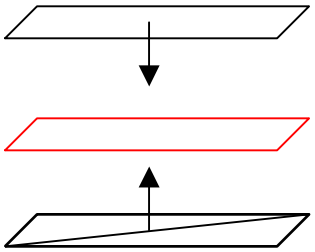


Figure 4.7: 2D on TIN condition

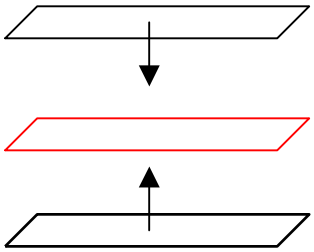


Figure 4.8: 2D on 2D condition

Of course there are many more topological situations (see chapter three), but for this thesis, at least all the topological situations discussed in the above part, have to be made possible with the model. How this is done will be presented in the next section.

4.3 The “Glue-Face” method

Conceptual model

The description of a new conceptual model, referred to as the Glue-Face model (fig 4.9), will be presented in this section. The Glue-Face model is based on the TIN-based Data Model and the 3D FDS Data Model, both discussed in section 4.1. The Glue-Face model has one entity, the “Face”. A Face is formed by three or more coordinate points, which can be seen as nodes or vertexes. The “Face” is used to form three elementary objects, area (2D), surface (2½ D) and body (volume) (3D). Finally, these different objects belong to different object classes. The primitive (Face), the three elementary objects (area, surface and body) and the object classes are the three fundamental levels of the Glue-Face model.

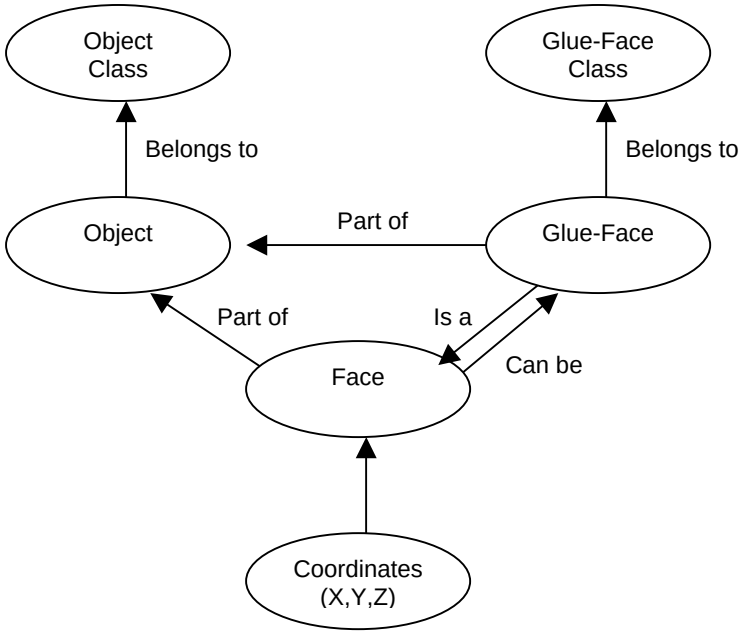


Figure 4.9: The proposed conceptual design of the Glue-Face model

In this conceptual model, the “ Glue-Face” is the key element and is used to overcome the topology problem of where an object is situated compared to another object. The Glue-Face is a face used to “glue” two faces together. A Glue-Face can be seen as double-sided tape (fig.4.10). In figure 4.10, the Glue-Face is visualised in red. Thanks to this Glue-Face, we know that object 2 is standing (is visualised) on object 1. A Glue-Face can be “in-bode” or “on-body” (fig.4.11). In figure 4.10, the Glue-Face is on-body.

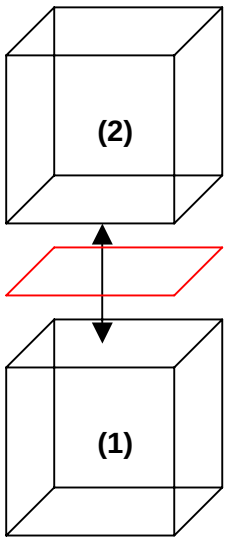


Figure 4.10: The “Glue-Face”

Figure 4.11: An “in-body” (A) and an “on-body” (B) “Glue-Face”

As mentioned, the Glue-Face model has one entity, the face. In this way, a great deal of the topological difficulty, mentioned in chapter three, is solved. This is because of the simple reason that relationships between nodes, edges, etc. or not necessary anymore, because these entities are not used (table 4.1). The only topology is the relationship between faces. This is done with the use of the Glue-Face. The face entity in the Glue-Face model is used to form 2D, 2½D and 3D objects. So in the model, a face can actually be a 2D, 2½D or a 3D face.

Therefore we use the following description:

2D Face: a 2D face has only x and y values (no z).
2½D Face: a 2½D face has x, y and z values, but can't be perpendicular in vertical direction.
3D Face: a 3D face has x, y and z values and can be in any direction.

So from now on we can say that the Glue-Face model is an integration model of (three) different types of faces.

2D/2½D entities	**3D entities**	**Glue-Face entities**
• connected nodes· • entity nodes • edges • faces	• connected nodes • entity nodes • space nodes • edges • faces • volumes	• faces
2D/2½D relationships	**3D relationships**	**Glue-Face relationships**
• start & end nodes • connected edges • containing face • contained entity nodes • left & right faces • outer & inner rings	• start & end nodes • connected edges • containing face • containing volume • contained entity nodes • contained space nodes • contained entity edges • bordered faces • bordered volumes • outer & inner rings • outer & inner shells	• order of nodes (comparable with start and end nodes) • glued faces

Table 4.1: Comparison of the topology concluded from chapter 3 with the Glue-Face topology.

The next stage in developing a 3D integration model is to convert the conceptual Glue-Face model into a logical model. This is described in the next section

Logical model

In the logical model phase, the translation of the conceptual model into a software dependent data structure is defined (see Appendix). Thus in this section the conceptual Glue-Face model will be translated into the logical Glue-Face model. Microsoft Access (an RDBMS) will be used for the implementation of the Glue-Face model; therefore the software dependent logical Glue-Face model will be a (object) relational model. Figure 4.12 shows the proposed logical design of the Glue-Face model.

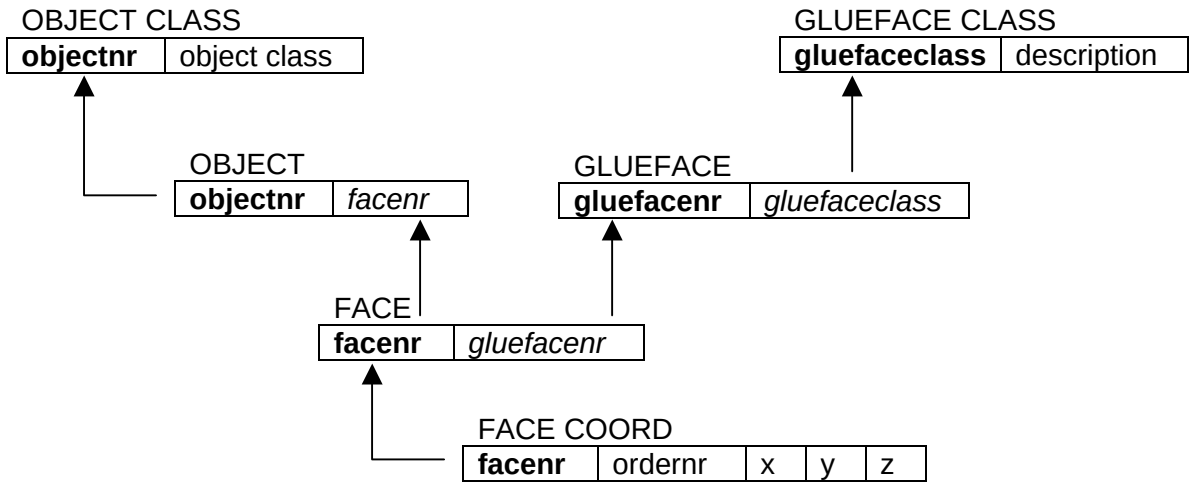


Figure 4.12: The proposed logical design of the Glue-Face model

In this logical model, the different relational tables are presented with their mutual relations. The relations are based on the following dependency statements:

1. An object, which is identified by an *objectnr* object number, belongs to one *objectclass* object class.
2. A glueface, which is identified by a *gluefacenr* glueface number, belongs to one *guefaceclass* glueface class.
3. A face, which is identified by a *facenr* face number, represents/is part of a 2D (area), 2½D (surface) or a 3D (body) object.
4. A face can be/have the function of glueface, identified by a *gluefacenr* glueface number.
5. A face is represented by face coordinates (x, y, z) with *ordernr* order number determining the order of the nodes/vertexes (begin and end node).

The logical Glue-Face model contains six relational tables, mentioned below a RT1 to RT6.

RT1: FACE COORD (**facenr**, ordernr, x, y, z)
RT2: FACE (**facenr**, *gluefacenr*)
RT3: GLUEFACE (**gluefacenr**, *gluefaceclass*)
RT4: OBJECT (**objectnr**, *facenr*)
RT5: GLUEFACE CLASS (**gluefaceclass**, description)
RT6: OBJECT CLASS (**objectnr**, objectclass)

The table names are given in capital letters. The primary key of every table is given in bold letters. Foreign keys are given in cursive letters. So, the table name of RT1 is FACE COORD. The primary key of RT2 is **facenr** and *facenr* is the foreign key of RT4.
The arrows in the logical Glue-Face model are representing many to one (n : 1) relationships between the relational tables.

The next stage is the implementation of the (logical) Glue-Face model in a DBMS. When the model is implemented in a DBMS it can be applicatited. The next chapter deals with the implementation and application of the Glue-Face model.

5 Implementation & Application

This chapter describes the implementation of the Glue-Face model in a DBMS and the application of the Glue-Face Model after implementation.

5.1 Implementation

The implementation of the Glue-Face model is done in Microsoft Access. For the implementation, first of all, a simple example scene was created (fig 5.1A and fig 5.1 B). This example scene contains simple 2D, 2½D and 3D objects, so that there can really be an integration of different types of faces that form these objects. With the creation of the example scene, the conditions discussed in chapter 4 have been taken into account. Most of the topological situations discussed in that part can be found in the example scene.

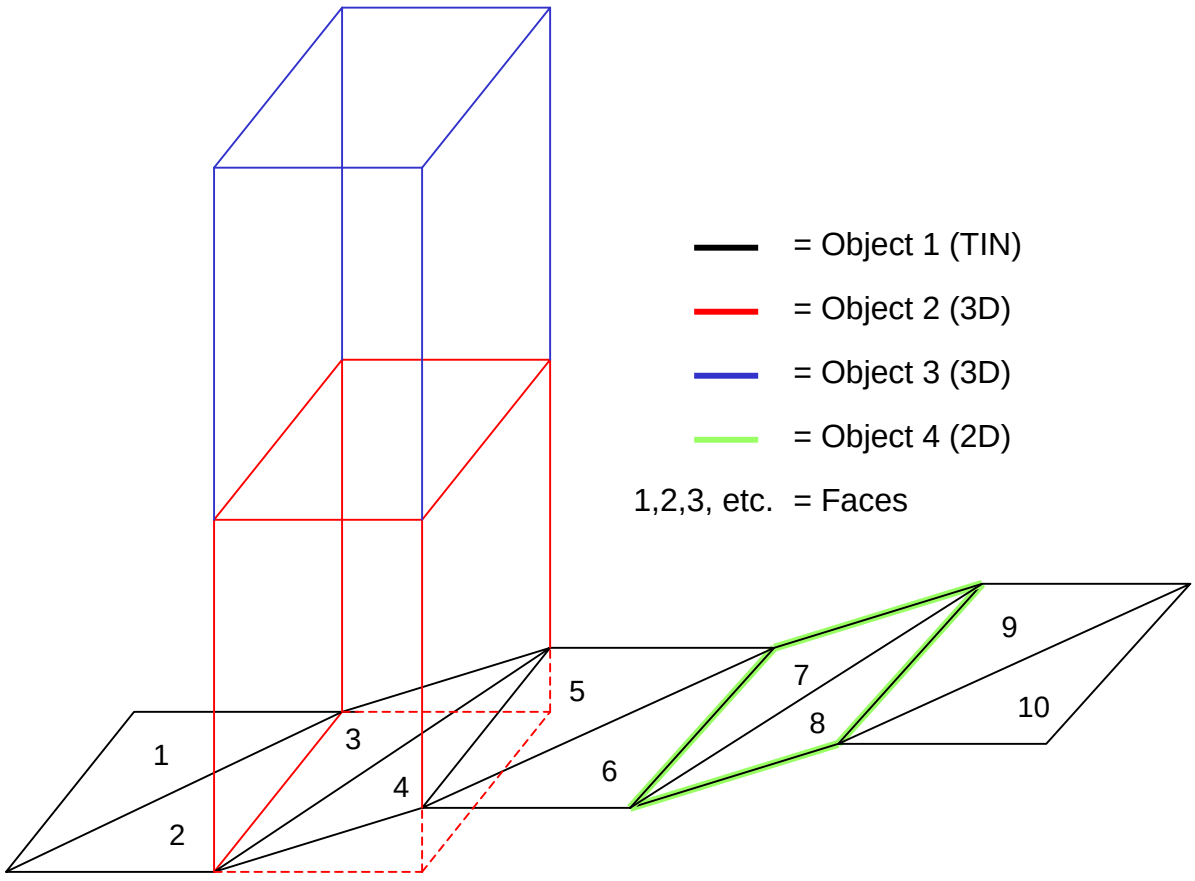


Figure 5.1 A: The example scene which is used for the implementation of the Glue-Face model.

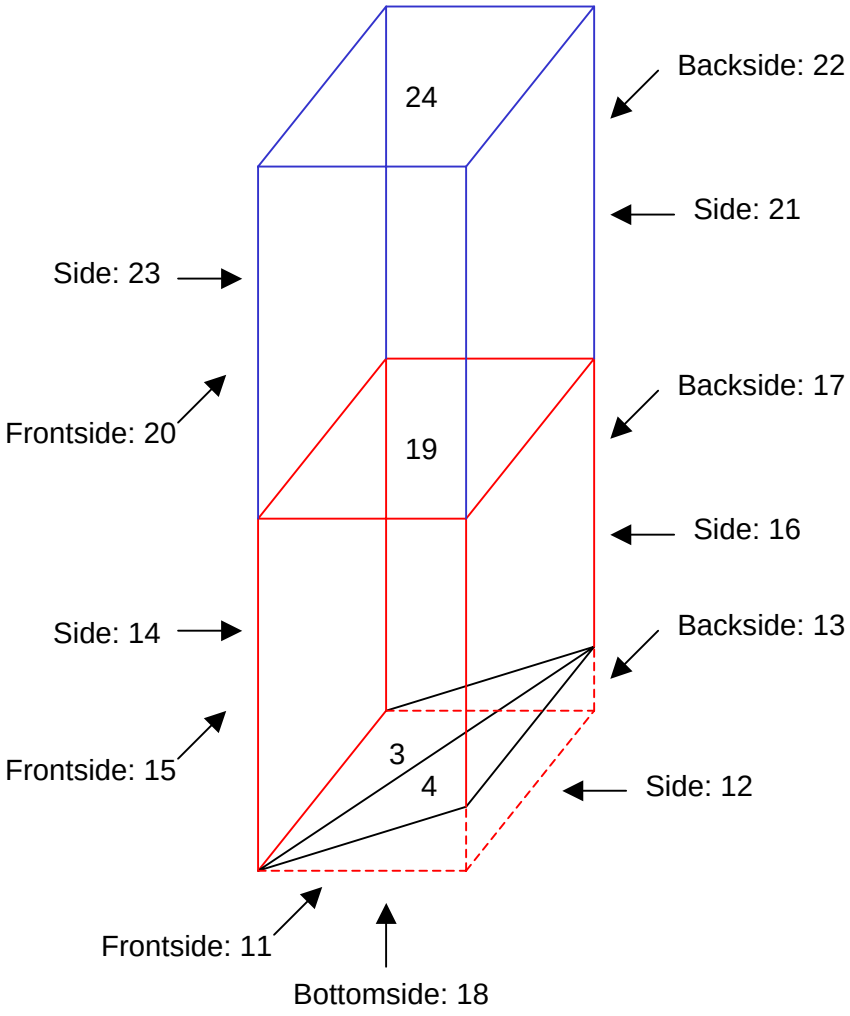


Figure 5.1 B: The example scene which is used for the implementation of the Glue-Face model.

For the implementation, this example scene is, based on the Glue-Face model, build in Microsoft Access. This leaded to the creation of the following six tables:

A. face_table
B. face_coord_table
C. object_table
D. object_type_table
E. glueface_table
F. glueface_type_table

The content of every table and what every single table means will be explained in the following part.

To begin with, the **face_table** (tab. 5.1 A) contains all the facenumbers (*face*) and the accompanying gluefacenumbers (*glueface*). This table is used to indicate if a certain face also has the function of glueface. As we look at figure 5.1.A. one can see that the facenumbers 2 and 3 together form gluefacenumber 1. Gluefacenumber – 9999 means that the accompanying facenumber is no glueface. So if a facenumber also has a gluefacenumber which is not - 9999, it means that that face also has the function of glueface.

In the **face_coord_table** (tab. 5.1 B) the coordinates (x, y and z values) of all the faces are stored. With *order*, the vertexes of the faces are indicated. So, as we look at figure 5.1 B, we see that *face* 1 is made of three vertexes with the coordinates (x1, y1, z1), (x2, y2, z2) and (x3, y3, z3). The “real” coordinates (not “x1” but for example “5”), which are used for the implementation, are chosen at own insight.

face_table	
face	glueface
1	-9999
2	1
3	1
4	1
5	-9999
6	-9999
…	…

Table 5.1 A: The face_table

face_coord_table				
face	order	x	y	z
1	1	x1	y1	z1
1	2	x2	y2	z2
1	3	x3	y3	z3
2	1	x4	y4	z4
2	2	x5	y5	z5
2	3	x6	y6	z6
…	…	…	…	…

Table 5.1 B: The face_coord_table

Which face(s) form a certain object is stored in the **object_table** (tab. 5.1 c).
We can see in this table that object 1 is formed out of the faces 1 to 10.

The **object_type_table** (tab 5.1 D) shows which kind of object type (TIN, 2d or 3d) the different objects are. In this case object 3 is a 3D object. With “TIN” a 2½ D object is meant.

object_table	
object	face
1	1
1	2
1	3
1	4
1	5
1	6
1	7
1	8
1	9
1	10
…	…

Table 5.1 C: The object_table

object_type_table	
object	object_type
1	tin
2	3d
3	3d
4	2d
…	…

Table 5.1 D: The object_type_table

In the **glueface_table** (tab. 5.1 E) is stored which type of glueface every glueface, earlier mentioned in the face_table (fig. 5.2 A), is. This is done with a glueface type number (*glueface_type*). So, glueface – 9999 is of type 2 and glueface 2 is of type 0.

Finally, the **glueface_type_table** (tab. 5.1 F), describes what every glueface type number means. There are three different kinds of gluefaces: *on_body*, *in_body* and *none*.
With on_body is meant that the glueface is situated "on" an object (at the outside) and in-body means that the glueface is inside an object. The description "none" means that the glueface is actually no glueface at all.

glueface_table	
glueface	glueface_type
-9999	2
1	1
2	0

Table 5.1 E: The glueface_table

glueface_type_table	
glueface_type	description
0	on_body
1	in_body
2	none

Table 5.1 F: The glueface_type_table

The visualisation of the coordinates which have been inputted in the **face_coord_table**, is shown in figure 5.2. As we look back at the example seen (fig. 5.1), one can see the resemblance.

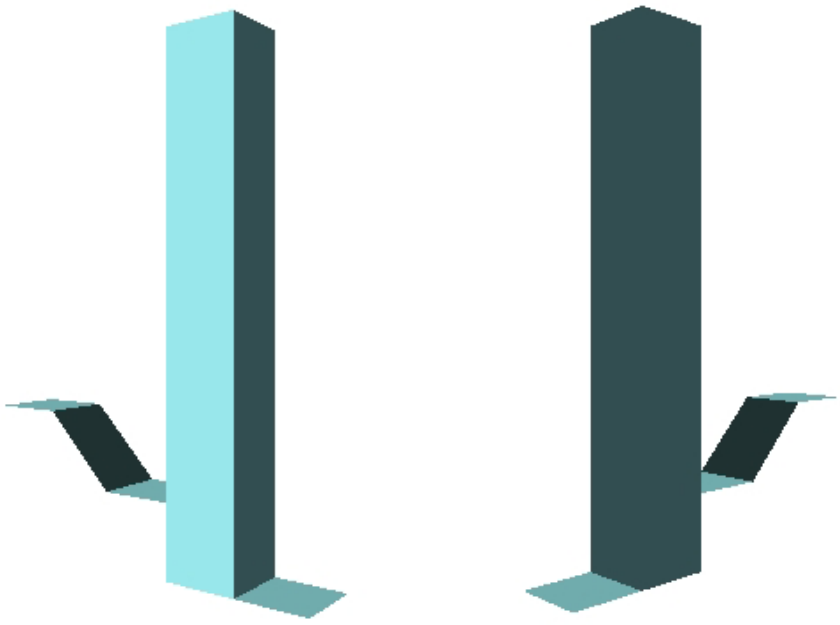


Figure 5.2: The visualisation of the coordinates in the face_coord_table

After the creation of the tables, the relations between the tables can be established, based on the Glue Face model. Figure 5.3 shows the Relational schema (model) as it was created in Microsoft Access. This model is known as the physical (or implementation) model (see Appendix). This model is a conversion of the logical model and describes the communication with the DBMS chosen for the implementation, in this with the RDBMS Microsoft Access.

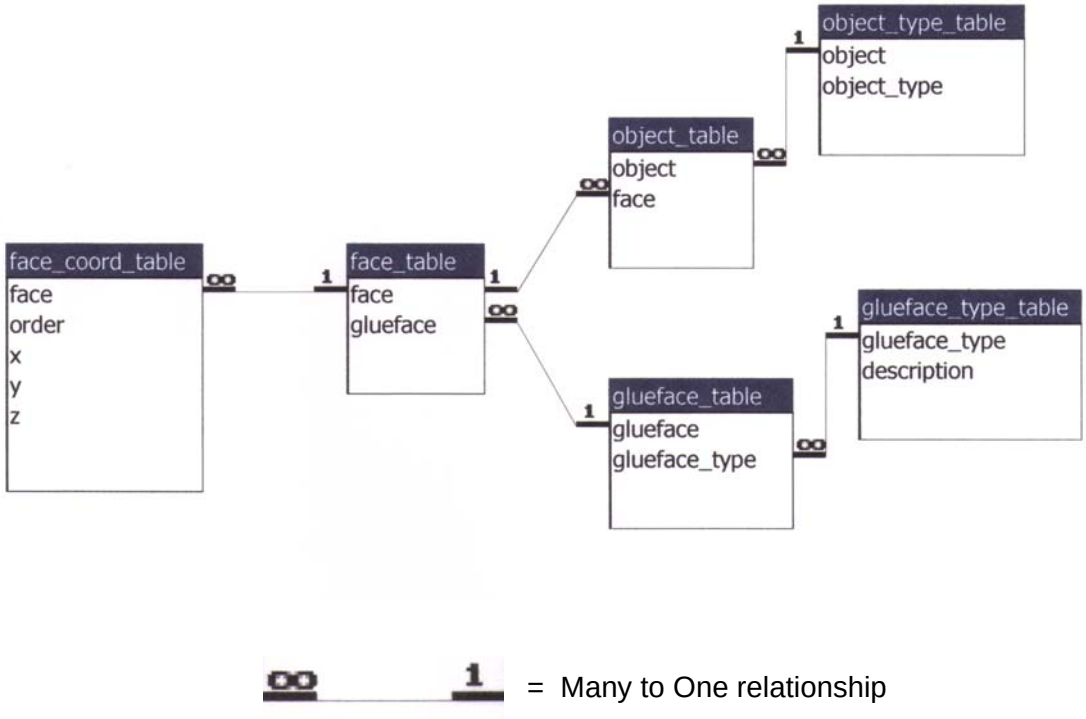


Figure 5.3: The Relational schema (model) in Microsoft Access, based on the logical Glue-Face model.

5.2 Model application

With model application, in this case, testing the Glue Face model is meant. This testing will be done by querying the created object-relational model based on the Glue Face model and the database of the example scene behind it.

Therefore, five simple queries are chosen:

1. Select object 2 (a 3D object)
2. Select the Glueface within object 2
3. Select the part of object 2 above the Glueface
4. Select the part of object 2 beneath the Glueface
5. Select object 4 (a 2D object)

These five queries comprise the most common (topological) queries. The results of the queries will be tables with coordinates. These coordinates will be visualised with the use of the visualization programme ArcScene. When the results of the queries are correct, the objective of this thesis, the integration of 2D, 2½D and 3D data, can be seen as succeeded.

The results of the five queries are presented below. The example scene (fig. 5.1) is placed below the result pictures, so they can be compared. The selections are in the result pictures visualised in red (due to shadow sometimes dark red).

Query 1: Select Object 2

SQL Query

```
DROP TABLE SELECTED_OBJECT;
CREATE TABLE SELECTED_OBJECT AS
SELECT DISTINCT B1.OBJECT, A1.FACE, A1.ORCOORD, A1.X, A1.Y, A1.Z
FROM FACE_COORD_TABLE1 A1, OBJECT_TABLE1 B1
WHERE B1.OBJECT = 2
AND B1.FACE = A1.FACE;


SELECT * FROM SELECTED_OBJECT
ORDER BY FACE, ORCOORD;
```

Result Table

OBJECT	FACE	ORCOORD	X	Y	Z
2	3	1	4	2	5
2	3	2	6	4	7
2	3	3	4	4	5
2	4	1	4	2	5
2	4	2	6	2	7
2	4	3	6	4	7
2	11	1	4	2	5
2	11	2	6	2	5
2	11	3	6	2	7
2	12	1	6	2	5
2	12	2	6	4	5
2	12	3	6	4	7
2	12	4	6	2	7
2	13	1	6	4	5
2	13	2	6	4	7
2	13	3	4	4	5
2	14	1	4	4	5
2	14	2	4	2	5
2	14	3	4	2	12
2	14	4	4	4	12
2	15	1	4	2	5
2	15	2	6	2	7
2	15	3	6	2	12
2	15	4	4	2	12
2	16	1	6	2	7
2	16	2	6	4	7
2	16	3	6	4	12
2	16	4	6	2	12
2	17	1	6	4	7
2	17	2	6	4	12
2	17	3	4	4	12
2	17	4	4	4	5
2	18	1	4	2	5
2	18	2	6	2	5
2	18	3	6	4	5
2	18	4	4	4	5
2	19	1	4	2	12
2	19	2	6	2	12
2	19	3	6	4	12
2	19	4	4	4	12

Result Pictures

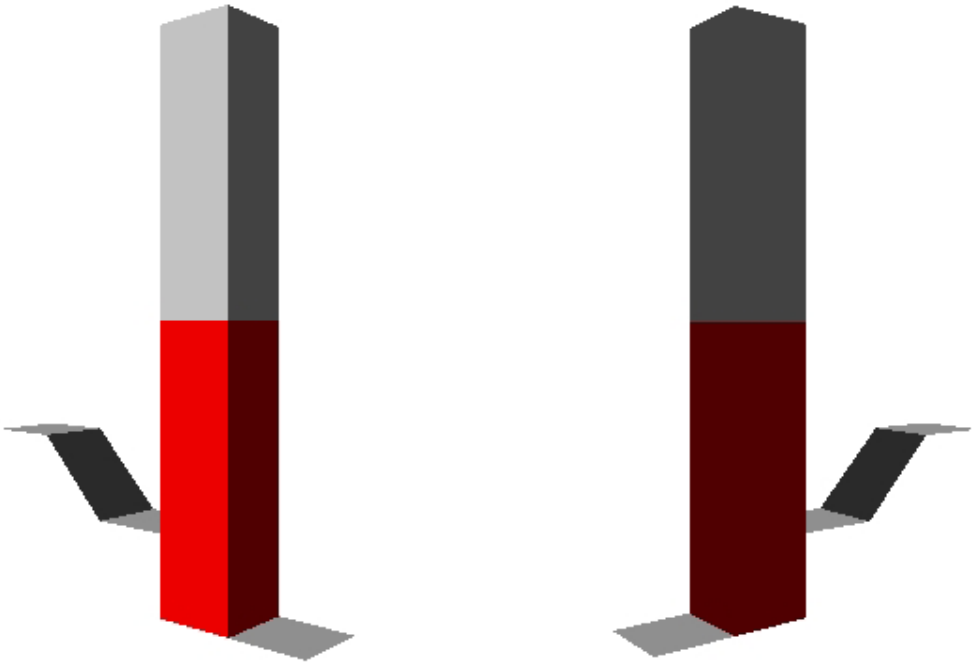
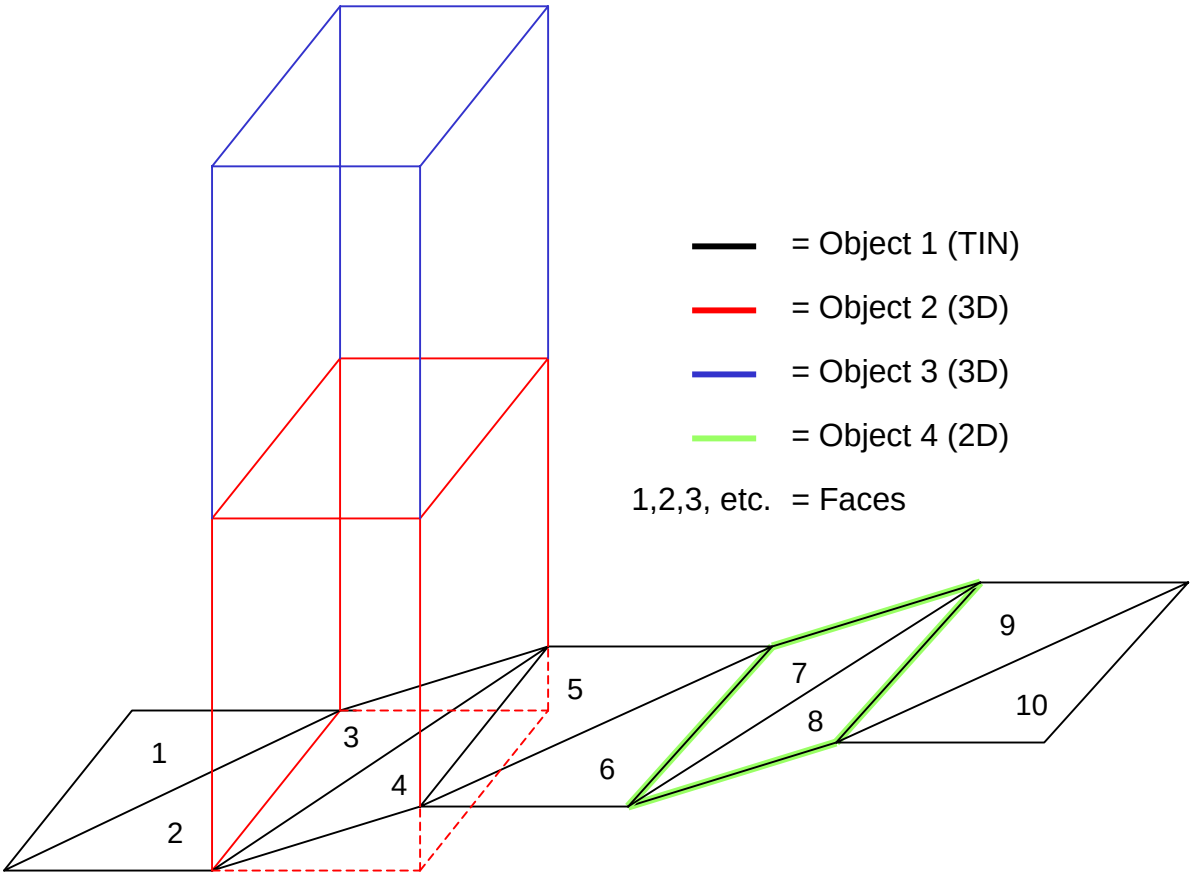


Figure 5.4: The selection of object 2



Query 2: Select the Glueface within object 2

SQL Query

```
DROP TABLE INNER_GLUEFACE;
CREATE TABLE INNER_GLUEFACE AS
SELECT B1.GLUEFACE, A1.FACE, A1.ORCOORD, A1.X, A1.Y, A1.Z
FROM SELECTED_OBJECT A1, FACE_TABLE B1, GLUE_FACE C1,
GLUEFACE_TYPE_TABLE D1
WHERE D1.GLUEFACE_TYPE = C1.GLUEFACE_TYPE
AND B1.GLUEFACE = C1.GLUEFACE
AND B1.FACE = A1.FACE
AND C1.GLUEFACE > 0
AND D1.DESCRIPTION = 'in_body';


SELECT * FROM INNER_GLUEFACE
ORDER BY FACE, ORCOORD;
```

Result Table

GLUEFACE	FACE	ORCOORD	X	Y	Z
1	3	1	4	2	5
1	3	2	6	4	7
1	3	3	4	4	5
1	4	1	4	2	5
1	4	2	6	2	7
1	4	3	6	4	7

Result Pictures

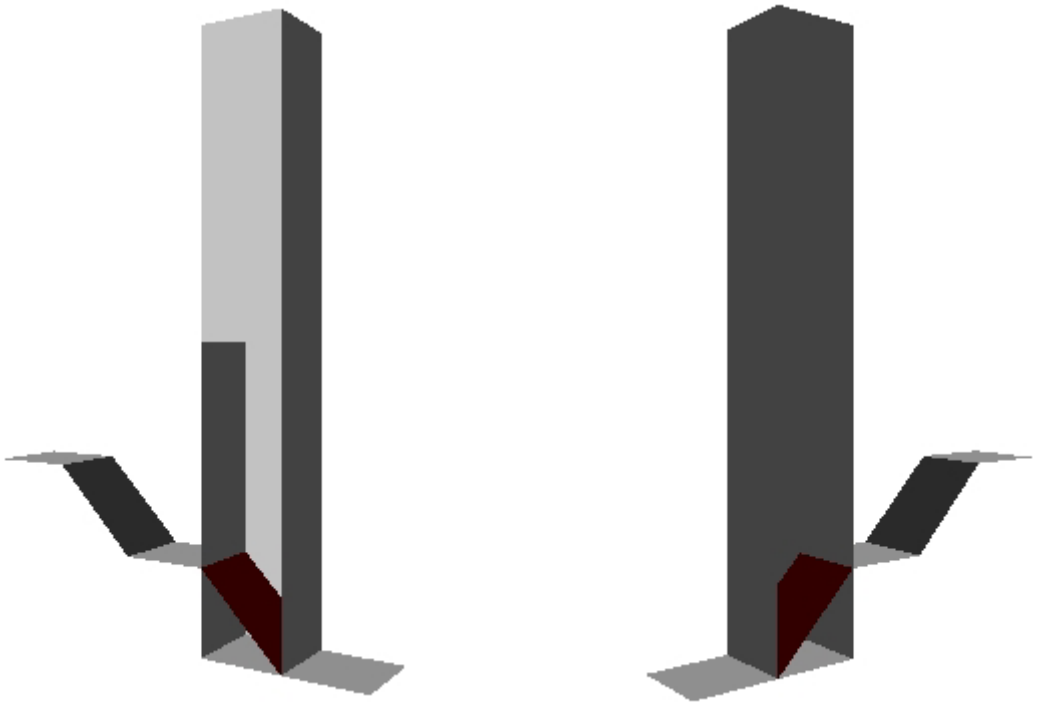
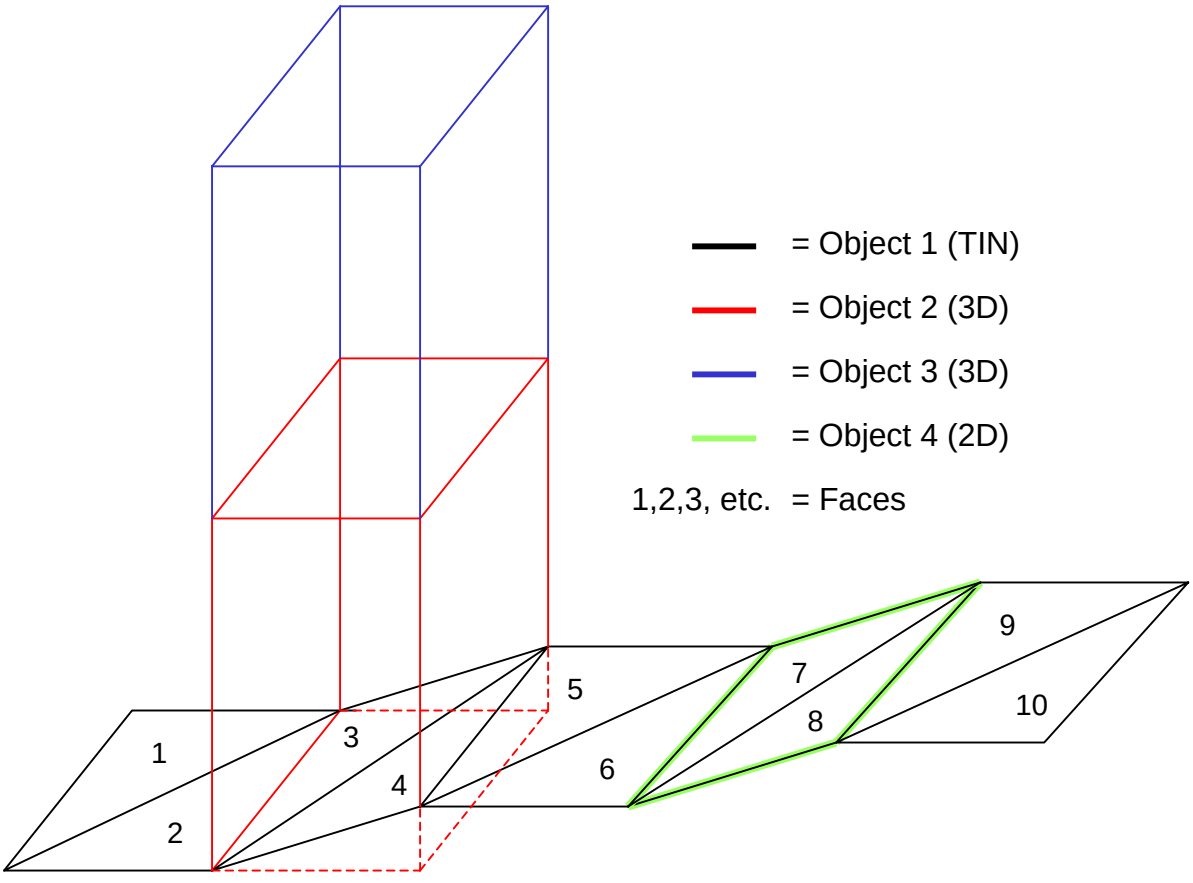


Figure 5.5: The selection of the Glueface within object 2



Query 3: Select the part of object 2 above the Glueface

SQL Query

```
DROP TABLE FACE_COUNT;
CREATE TABLE FACE_COUNT (FACE, FACE_COUNT) AS
SELECT FACE, COUNT (ORCOORD)
FROM FACE_COORD_TABLE1
GROUP BY FACE;

DROP TABLE SELECTED_OBJECT;
CREATE TABLE SELECTED_OBJECT AS
SELECT DISTINCT B1.OBJECT, A1.FACE, A1.ORCOORD, A1.X, A1.Y, A1.Z
FROM FACE_COORD_TABLE1 A1, OBJECT_TABLE1 B1
WHERE B1.OBJECT = 2
AND B1.FACE = A1.FACE;

DROP TABLE INNER_GLUEFACE;
CREATE TABLE INNER_GLUEFACE AS
SELECT B1.GLUEFACE, A1.FACE, A1.ORCOORD, A1.X, A1.Y, A1.Z
FROM SELECTED_OBJECT A1, FACE_TABLE B1, GLUE_FACE C1,
GLUEFACE_TYPE_TABLE D1
WHERE D1.GLUEFACE_TYPE = C1.GLUEFACE_TYPE
AND B1.GLUEFACE = C1.GLUEFACE
AND B1.FACE = A1.FACE
AND C1.GLUEFACE > 0
AND D1.DESCRIPTION = 'in_body';

DROP TABLE UPPER_PART_TMP;
CREATE TABLE UPPER_PART_TMP AS
SELECT DISTINCT A1.FACE, A1.ORCOORD, A1.X, A1.Y, A1.Z
FROM SELECTED_OBJECT A1, INNER_GLUEFACE B1
WHERE (A1.X = B1.X AND A1.Y = B1.Y AND A1.Z >= B1.Z);

DROP TABLE SEL_UPPER_FACE_COUNT;
CREATE TABLE SEL_UPPER_FACE_COUNT (FACE, FACE_COUNT) AS
SELECT FACE, COUNT (ORCOORD)
FROM UPPER_PART_TMP
GROUP BY FACE;

DROP TABLE UPPER_PART;
CREATE TABLE UPPER_PART AS
SELECT DISTINCT A1.FACE, A1.ORCOORD, A1.X, A1.Y, A1.Z
FROM UPPER_PART_TMP A1, UPPER_PART_TMP B1, SEL_UPPER_FACE_COUNT C1,
FACE_COUNT D1
WHERE A1.FACE = B1.FACE
AND B1.FACE = C1.FACE
AND C1.FACE = D1.FACE
AND C1.FACE_COUNT = D1.FACE_COUNT;

SELECT * FROM UPPER_PART
ORDER BY FACE, ORCOORD;
```

Result Table

FACE	ORCOORD	X	Y	Z
3	1	4	2	5
3	2	6	4	7
3	3	4	4	5
4	1	4	2	5
4	2	6	2	7
4	3	6	4	7
14	1	4	4	5
14	2	4	2	5
14	3	4	2	12
14	4	4	4	12
15	1	4	2	5
15	2	6	2	7
15	3	6	2	12
15	4	4	2	12
16	1	6	2	7
16	2	6	4	7
16	3	6	4	12
16	4	6	2	12
17	1	6	4	7
17	2	6	4	12
17	3	4	4	12
17	4	4	4	5
19	1	4	2	12
19	2	6	2	12
19	3	6	4	12
19	4	4	4	12

Result Pictures

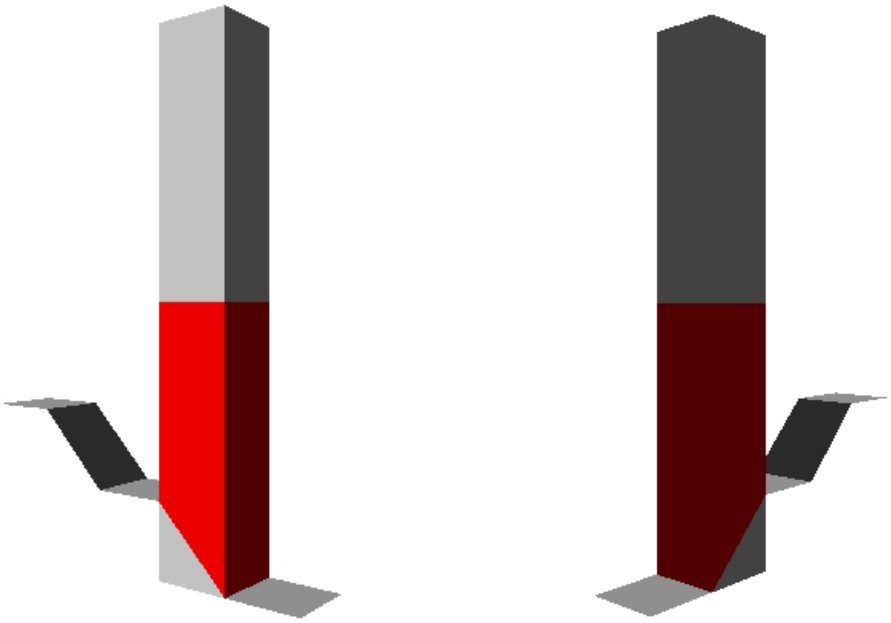
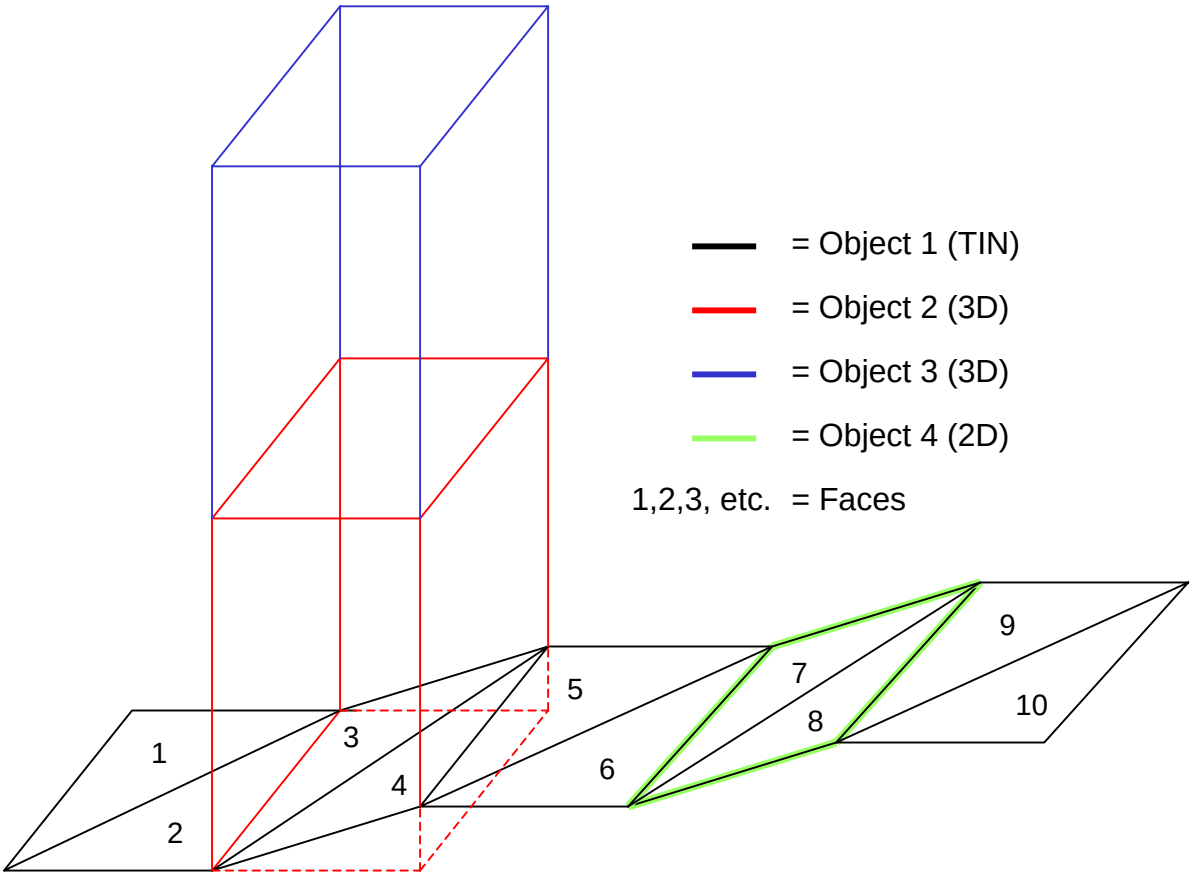


Figure 5.6: The selection of the part of object 2 above the Glueface



Query 4: Select the part of object 2 beneath the Glueface

SQL Query

```
DROP TABLE FACE_COUNT;
CREATE TABLE FACE_COUNT (FACE, FACE_COUNT) AS
SELECT FACE, COUNT (ORCOORD)
FROM FACE_COORD_TABLE1
GROUP BY FACE;

DROP TABLE SELECTED_OBJECT;
CREATE TABLE SELECTED_OBJECT AS
SELECT DISTINCT B1.OBJECT, A1.FACE, A1.ORCOORD, A1.X, A1.Y, A1.Z
FROM FACE_COORD_TABLE1 A1, OBJECT_TABLE1 B1
WHERE B1.OBJECT = 2
AND B1.FACE = A1.FACE;

DROP TABLE INNER_GLUEFACE;
CREATE TABLE INNER_GLUEFACE AS
SELECT B1.GLUEFACE, A1.FACE, A1.ORCOORD, A1.X, A1.Y, A1.Z
FROM SELECTED_OBJECT A1, FACE_TABLE B1, GLUE_FACE C1,
GLUEFACE_TYPE_TABLE D1
WHERE D1.GLUEFACE_TYPE = C1.GLUEFACE_TYPE
AND B1.GLUEFACE = C1.GLUEFACE
AND B1.FACE = A1.FACE
AND C1.GLUEFACE > 0
AND D1.DESCRIPTION = 'in_body';

DROP TABLE LOWER_PART_TMP;
CREATE TABLE LOWER_PART_TMP AS
SELECT DISTINCT A1.FACE, A1.ORCOORD, A1.X, A1.Y, A1.Z
FROM SELECTED_OBJECT A1, INNER_GLUEFACE B1
WHERE (A1.X = B1.X AND A1.Y = B1.Y AND A1.Z <= B1.Z);

DROP TABLE SEL_LOWER_FACE_COUNT;
CREATE TABLE SEL_LOWER_FACE_COUNT (FACE, FACE_COUNT) AS
SELECT FACE, COUNT (ORCOORD)
FROM LOWER_PART_TMP
GROUP BY FACE;

DROP TABLE LOWER_PART;
CREATE TABLE LOWER_PART AS
SELECT DISTINCT A1.FACE, A1.ORCOORD, A1.X, A1.Y, A1.Z
FROM LOWER_PART_TMP A1, LOWER_PART_TMP B1, SEL_LOWER_FACE_COUNT C1,
FACE_COUNT D1
WHERE A1.FACE = B1.FACE
AND B1.FACE = C1.FACE
AND C1.FACE = D1.FACE
AND C1.FACE_COUNT = D1.FACE_COUNT;

SELECT * FROM LOWER_PART
ORDER BY FACE , ORCOORD;
```

Result Table

FACE	ORCOORD	X	Y	Z
3	1	4	2	5
3	2	6	4	7
3	3	4	4	5
4	1	4	2	5
4	2	6	2	7
4	3	6	4	7
11	1	4	2	5
11	2	6	2	5
11	3	6	2	7
12	1	6	2	5
12	2	6	4	5
12	3	6	4	7
12	4	6	2	7
13	1	6	4	5
13	2	6	4	7
13	3	4	4	5
18	1	4	2	5
18	2	6	2	5
18	3	6	4	5
18	4	4	4	5

Result Pictures

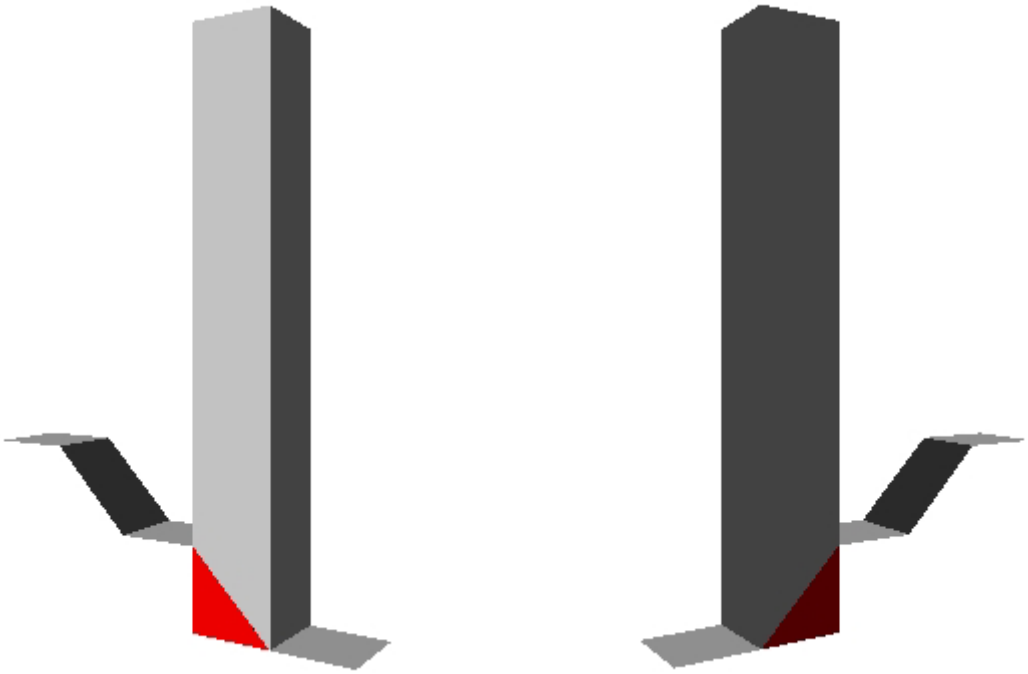
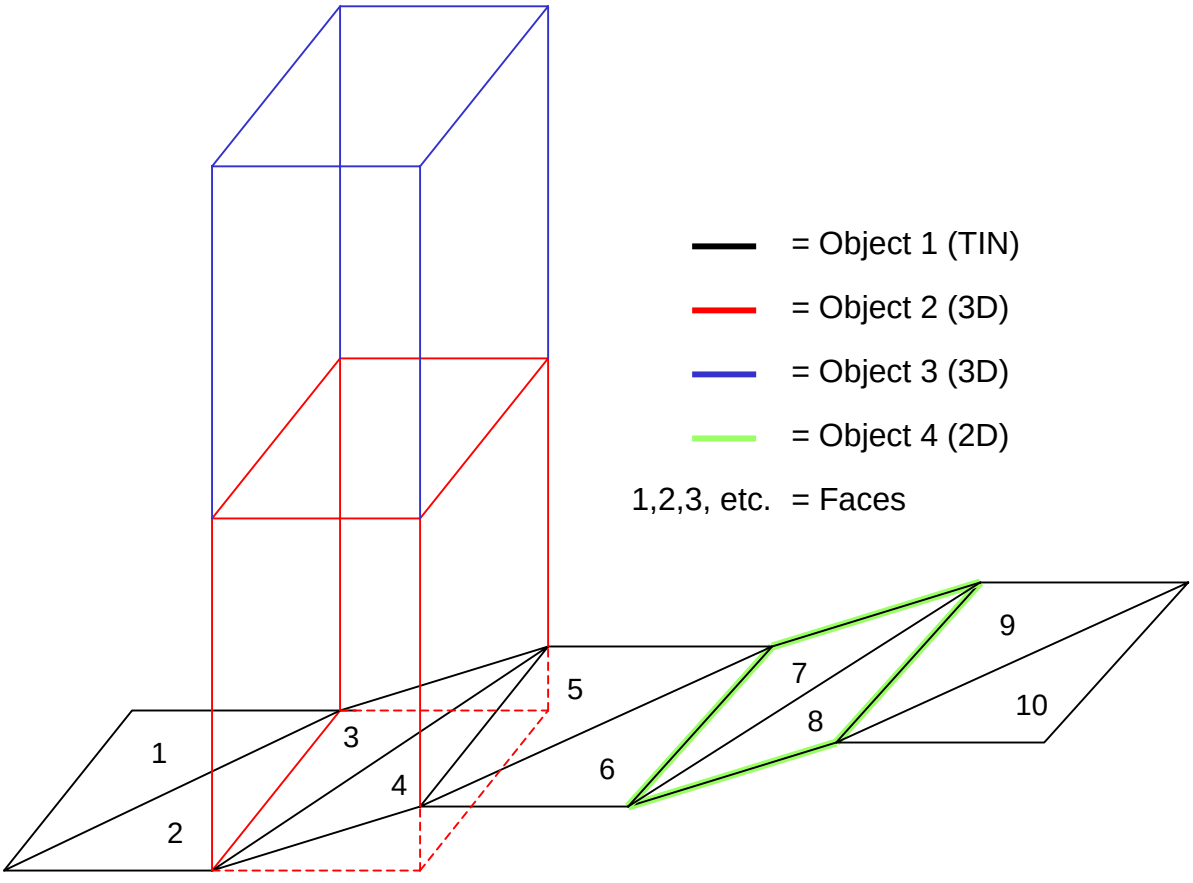


Figure 5.7: The selection of the part of object 2 beneath the Glueface



Query 5: Select object 4

SQL Query

```
DROP TABLE SELECTED_2D_OBJECT;
CREATE TABLE SELECTED_2D_OBJECT AS
SELECT DISTINCT B1.OBJECT, A1.FACE, A1.ORCOORD, A1.X, A1.Y, A1.Z
FROM FACE_COORD_TABLE1 A1, OBJECT_TABLE1 B1
WHERE B1.OBJECT = 4
AND A1.FACE = B1.FACE;

SELECT * FROM SELECTED_2D_OBJECT;
ORDER BY FACE , ORCOORD;
```

Result Table

OBJECT	FACE	ORCOORD	X	Y	Z
4	7	1	8	2	7
4	7	2	10	4	9
4	7	3	8	4	7
4	8	1	8	2	7
4	8	2	10	2	9
4	8	3	10	4	9

Result Pictures

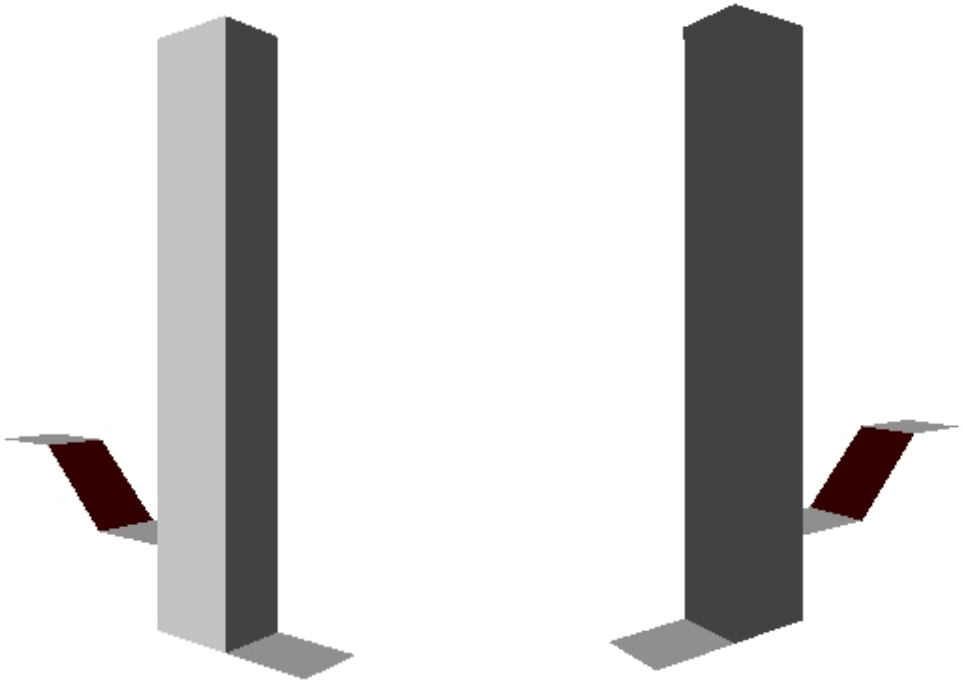
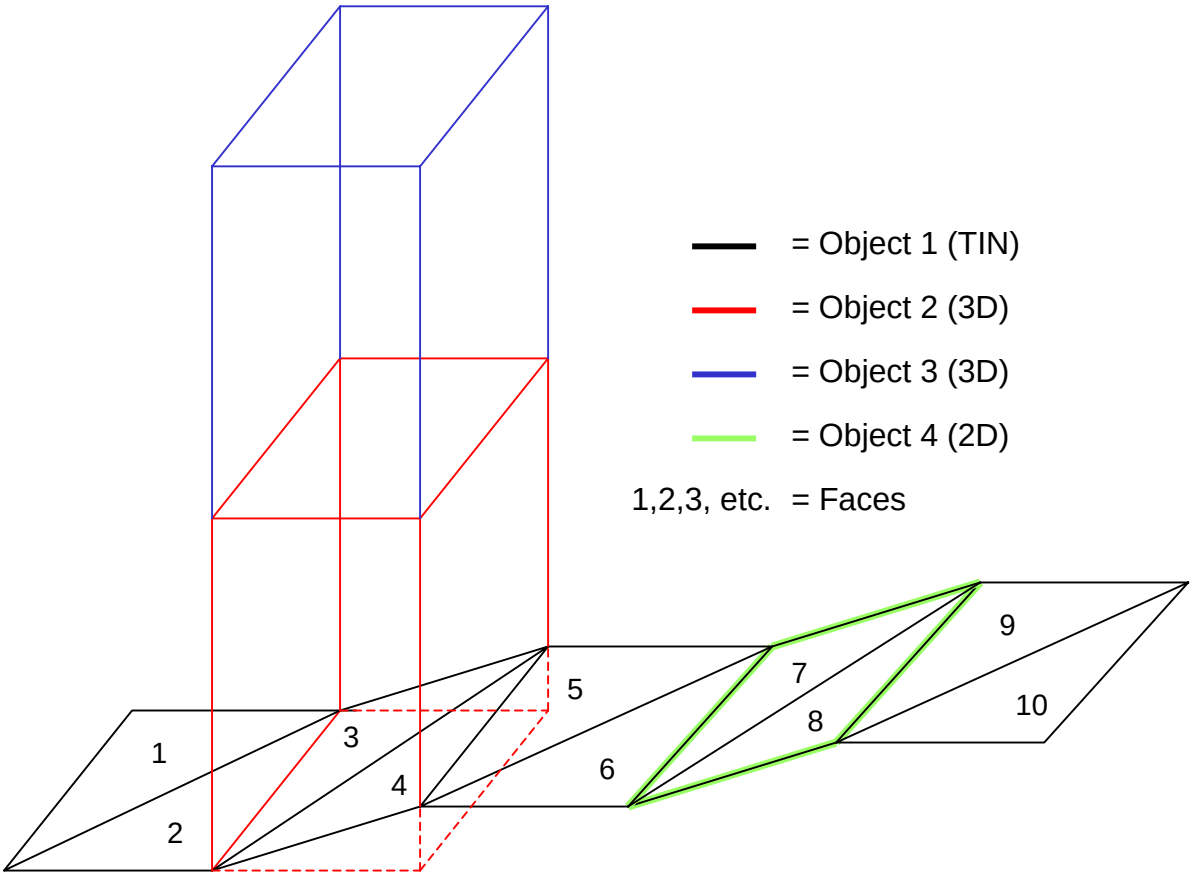


Figure 5.8: The selection of object 4



5.3 Remarks

The rather complicated queries are not possible in Microsoft Access. So, for this research, the Access tables first had to be converted to Oracle Spatial tables, because Oracle Spatial can handle such queries. The outcome of the queries, tables with coordinates of the selected objects, had to be visualized. Therefore the Oracle Spatial tables, again, had to be converted to a format which could be read by ArcScene, like shapefile format (.shp). This conversion is done with the use of FME (Feature Manipulation Engine). FME can convert spatial data from one format to another. This conversion, eventually leaded to the end result, ArcScene pictures.

5.4 Conclusion

As we compare the results of the five queries (tables and pictures) with the original example scene, we can see that with every query, the correct faces have been selected. So therefore we can conclude that, at least for the simple example scene, the integration of 2D, 2½D and 3D data with the Glue Face Model works.

6 Conclusion, discussion and further research

This thesis is focussing on the development of a 3D integration model. This is just a small, but important part in the development of 3D GIS. In the following part the outcomes of the thesis will be discussed and some directions for further research will be given.

6.1 Conclusion

The objective of this thesis is to integrate geo data sets that describe, in a 2, 2½ and 3 dimensional way, real world objects. This is done by developing a 3D integration model. As was concluded in chapter three, the biggest problem in developing a 3D integration model is the difference in topology. Therefore the Glue-Face model is designed. The Glue-Face model uses gluefaces to overcome the topological problem. A glueface is a face where two objects are “glued” together. The Glue-Face model is partly based on the 3D FDS model (Molenaar, 1990) and the TIN-based model (Pilouk, 1996).
After the development of the Glue-Face model, the model is implemented. This implementation is necessary to be able to test the model, to check if it meets the expectations. Testing the model is done by querying the model. Therefore, five simple selection queries are formulated. If the outcomes of the queries are correct, the objective of the thesis is achieved. In chapter five, the outcomes of the queries are shown. As we can see, all the outcomes are correct. So we may conclude that the Glue-Face model is able to integrate 2D, 2½D and 3D data, and therefore that the goal of this thesis is achieved.

6.2 Discussion

In this part the extension of the 3D FDS Data Model with the Glue-Face model is discussed.

Extension 3D FDS Data Model

The 3D FDS Data Model (fig. 6.1) is a data model that describes full 3D topology (see also chapter 4.1, figure 4.2). The Glue-Face Model (fig. 6.2) can be integrated with the 3D FDS Data Model. Figure 6.3 shows the 3D FDS Data Model with the integrated Glue-Face model. The Glue-Face model is used as an extension of the 3D FDS Data Model. In this way, the 3D FDS Data Model can also be used for simply integrating 2D, 2½D and 3D faces. This can be an outcome for situations that don’t require the extensive full 3D topology, like planning applications used for simple queries, estimations and visualisations. In this way, with the Glue-Face extension, time and effort can be saved, because in some cases the extensive full 3D topology is not needed and used. This extension of the 3D FDS Data Model with the Glue-Face model is still theoretical and needs further research.

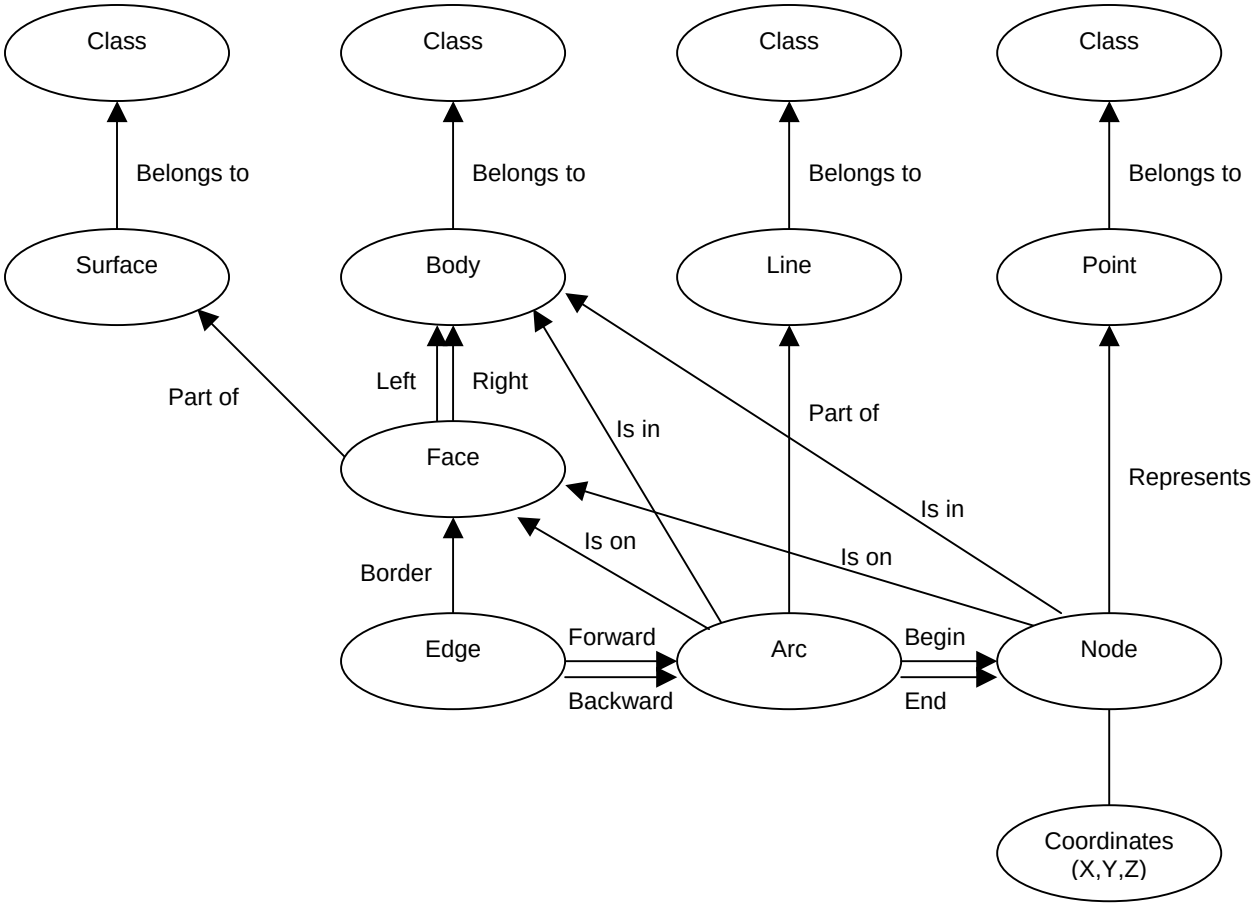


Figure 6.1: 3D Formal Data Structure (FDS) Data Model (after Molenaar 1990)

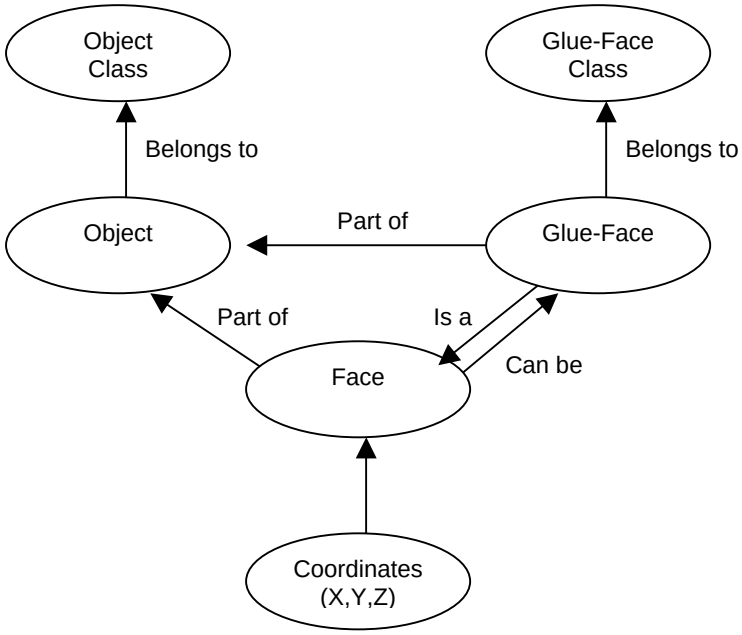


Figure 6.2: The Glue-Face Model

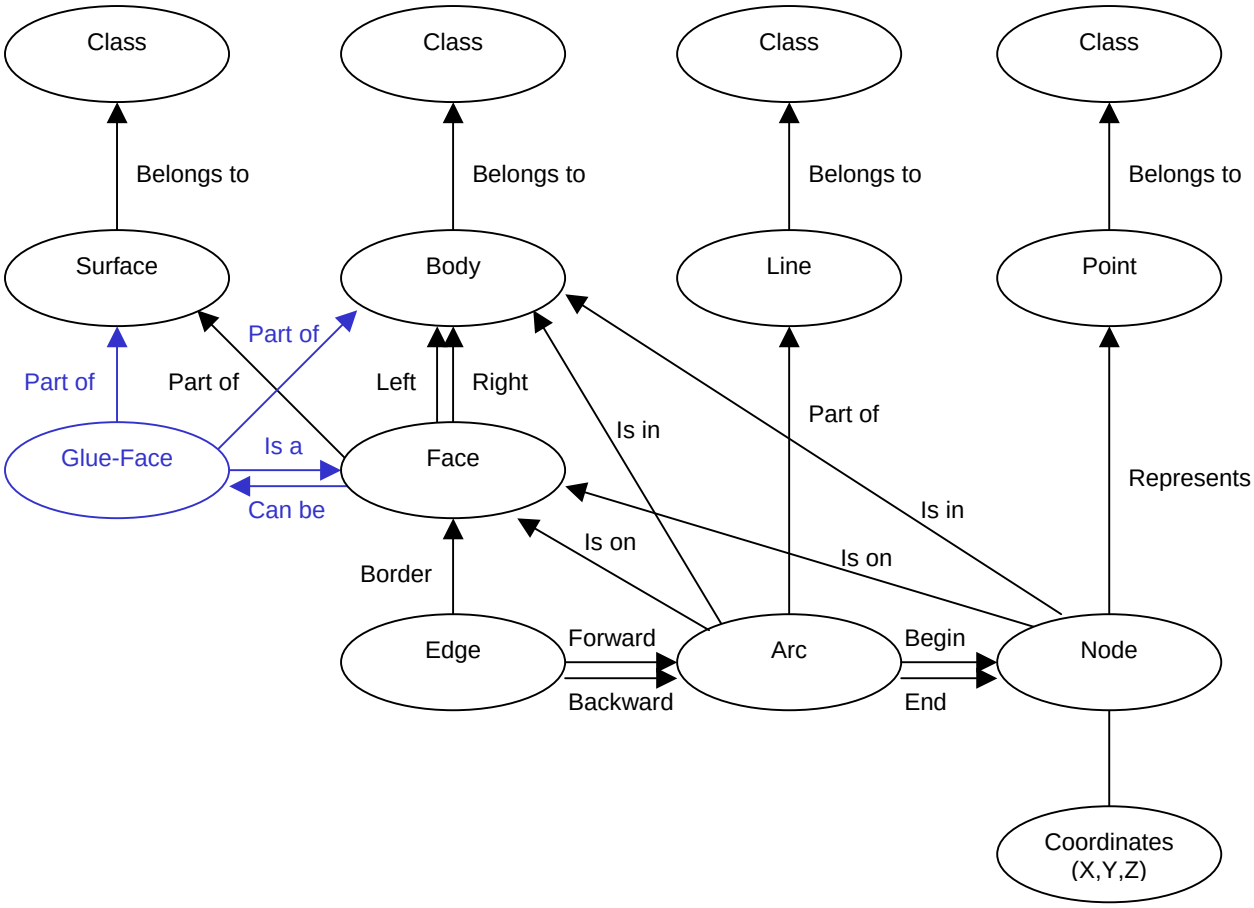


Figure 6.3: 3D Formal Data Structure (FDS) Data Model (after Molenaar 1990), with the Glue-Face extension (visualised in blue)

6.3 Further research

In this section some suggestions for further research are given. These suggestions stretch upon the Glue-Face model.

Simplified model

The Glue-Face model is a simple model. The model has only one primitive, the face. So only objects with faces can be used in the model. The model cannot handle points and lines. The model would be applicable on a more extensive scale if it could also handle points and lines. One could thing of an extension of the Glue-Face model and change the model into the “Glue-Entity” model (GEM). For this thesis, just for developing the Glue-Face method, the Glue-Face model satisfies.

Constrains and editing rules

There have to be thought of more constrains and editing rules for the Glue-Face mode. In this thesis is worked with a test database designed for this thesis. For example, If you want to add data (objects) or delete data from the database, there have to be some editing rules, otherwise working with the model wont be possible anymore. This is a pretty difficult technical aspect. And there are many more situations one can think of, were not yet constrains or editing rules have been designed for.

References

[ADA] N. R. Adam, A. Gangopadhyay (1997). Database issues in geographic information systems, ISBN 0-7923-9924-2, Dordrecht, The Netherlands.

[BUR] P.A. Burrough, R.A. McDonnell (1998). Principles of Geographical Information Systems, ISBN 0-19-823366-3, New York, USA.

[CUB] S. Cuberes Farreny (2002). DEMs in Different Geo-Information Environments, Thesis report GIRS-2002-28, Wageningen, The Netherlands.

[ESRI] ESRI (2002). ArcView 3.2a, 3D Analyst. URL: www.esri.com

[DÖL] J. Döllner, K. Hinrichs (2000). An Object-Oriented Approach for Integrating 3D Visualization Systems and GIS, Computer & Geosciences, 26 (2002), 67-76, Elsevier Science Verlag.

[HEA] H.M. Hearnshaw, D.J. Unwin (1994). Visualization in Geographical Information Systems, ISBN 0-471-94435-1, West Sussex, England.

[MOL] M. Molenaar (1998). An Introduction to the Theory of Spatial Object Modelling for GIS, ISBN 0-7484-0775-8, London, England.

[MUR] S. Murai (1996). GIS Work Book (Fundamental Course). October, 1996
URL: www.profc.udec.cl/~gabriel/tutoriales/giswb/vol1/contents.htm

[KOF] M. Kofler (1998). R-trees for Visualizing and Organizing Large 3D GIS Databases Graz, July 1998.
URL: www.icg.tu-graz.ac.at/kofler/thesis

[KWA] M. Kwan, J. Lee (2003). Geovisualization of Human Activity Patterns Using 3D GIS: A Time-Geographic Approach. In Michael F. Goodchild and Donald G. Janelle. EDS. 2003. Spatially Integrated Social Science: Examples in Best Practice, Chapter 3. Oxford University Press.

[KOV] S. Kovtchega. Aspects of Spatial Visualization in GIS, Literature Review.
GIS Set B A00527270

[OOS1] P.J.M. van Oosterom (1990). Reactive Data Structures for Geographic Information Systems, December 1990.

[OOS2] P.J.M. van Oosterom, J.E. Stoter (2002). Incorporating 3D geo-objects into a 2D geo-DBMS, FIG XXII/ACSM-ASPRS, Washington, D.C. USA, April 19-26 2002.

[OOS3] P.J.M. van Oosterom, J.E. Stoter, M. A. Salzmann and P. van der Molen (2002). Towards a 3D Cadastre, FIG XXII International Congress, Washington, D.C. USA, April 19-26 2002.

[OOS4] P.J.M. van Oosterom, J.E. Stoter, S. Zlatanova, W. Quak (2002). The balance between Geometry and Topology.
URL: www.geo.tudelft.nl/frs/staff/sisi/thesis/html/refer/ps/topo_geom02.pdf

[OOS5] P.J.M van Oosterom, J.E. Stoter, S. Zlatanova, E. Verbree (2002). Onderzoek brengt 3D GIS in gangbare geo-software naderbij, ViMATRIX, Page 26-29, September 2002.

[OOS6] P.J.M. van Oosterom, J.E. Stoter (2002). Een oplossing voor een 3D-kadaster?, GEODESIA, page 288-296, 2002-7/8.

[ORACLE] Oracle (2002). Oracle Spatial 9iR2 technical white paper.
URL: www.oracle.com

[PIL] M. Pilouk (1996). Integrated Modelling for 3D GIS, ISBN 90-6164-122-5, Enschede, The Netherlands.

[SAN] D.C. Sandison (1999). Using a Four-Dimensional Geographical Information System to Visualise the Environmental Impact of Smog. Curtin University of Technology, July 1999.
URL: http://www.users.bigpond.com/dsandison/

[SAR] F. Sarkozy (2001). Some Remarks on the Development of the Spatial Data Model. Technical University Budapest, November 2001.
URL: http://bme-geod.agt.bme.hu/public_e/mod/datamodel.htm

[SUN] M. Sun, J. Li, N. Jing (1999). Spatial Database Techniques Oriented to Visualization in 3D GIS.
URL: www.digitalearth.ca/html_papers/DE_A_064.htm

[SHI] Y. Shimura, S. Huang, R. Shibasaki (2000). A Surface Interpolation for Large-scale Representation of Terrain in An Urban Area.
URL: www.gisdevelopment.net/aars/acrs/2000/ts7/gdi008.shtml

[STO] J. E. Stoter, J. A. Zevenbergen (2001). Changes in the definition of property: A consideration for a 3D cadastral registration system. FIG Working Week 2001, Seoul, Korea, 6-11 May 2001.

[SWA] J. Swanson (1996). The Three Dimensional Visualization & Analysis of Geographic Data, 1996.
URL: http://maps.unomaha.edu/Peterson/gis/Final_Projects/1996/Swanson/GIS_Paper.html

[TRO] K.C. Trott, I. Greasley (1999). A 3D spatial data model for terrain reasoning. PAR Government Systems Corporation
URL: www.geovista.psu.edu/sites/geocomp99/Gc99/037/gc_037.htm

[VRML] VRML (1997). VRML Consortium Inc.: The Virtual Reality Modelling Language; International Standard ISO/IEC 14772-1:1997.
URL: www.vrml.org

[WAT] R.T. Watson (2002). Data Management: Databases and Organizations (Third Edition), ISBN 0-471-41845-5, West Sussex, England.

[ZLAT1] S. Zlatanova (2000). 3D GIS for Urban Development, ISBN 90-6164-178-0, Graz, Austria, March 2000.

[ZLAT2] S. Zlatanova, A.A. Rahman, M. Pilouk (2002). 3D GIS: Current status and perspectives.
URL: www.geo.tudelft.nl/frs/staff/sisi/thesis/html/refer/ps/SZ_AR_MP02.pdf

[ZLAT3] S. Zlatanova, A.A. Rahman. M. Pilouk (2000). The 3D GIS Software Development: global efforts from researchers and vendors.
URL: www.geo.tudelft.nl/frs/staff/sisi/thesis/html/refer/ps/AR_MP_SZ01.pdf

GIS Glossary's:

[GLOS1] http://www.esri.com/library/glossary/glossary.html

[GLOS2] http://www.cadforum.com/gisglossary

[GLOS3] http://www.urbansimulation.com/fea_glossary.php

[GLOS4] http://www.auf.uni-rostock.de/gg/cebit_e/glossar_e.html

Appendix: Principles of Data Modelling

This appendix tries to explain the most common data modelling methodologies and terms. Well-defined data models offer a simple way (by querying) to answer questions or solve problems. Thus a data model is mostly application dedicated.

Principles of data modelling

In data modelling, three terms are often used: *conceptual model*, *logical model* and *physical model*. These terms refer to different phases in the modelling process with different levels of detail. The conceptual model is used to identify which objects are needed in the model with their characteristics and relationships. The conceptual model is independent of the database management system (DBMS) used for system implementation [ADA]. In the logical model phase, the translation of the conceptual model into a software dependent data structure is defined. The last phase, the physical (or implementation) model, is a conversion of the logical model and describes the communication with the DBMS chosen for the implementation.

Conceptual Data Model

The Entity-Relationship model is on of the most popular and widely used conceptual data models. The ER model uses *entities*, *relationships* and *attributes*. Entities can be objects are processes. The characteristics of an entity are called attributes. Relationships are used to describe the interaction or association between entities and can be *one-to-one*, *one-to-many*, *many-to-one* or *many-to-many*, depending on the number of participating entities.
An attribute which can be used for the unique identification of an entity or a relationship is called a *primary key*. If there are more attributes, which are potential primary keys, these attributes are called *candidate keys*. A group of attributes together can also form a unique identifier, known as a *composite key*.

Logical Data Model

From the several logical data models, like the *relational*, *hierarchical* and *network* models, the relational model has become the dominant model for database applications. The *relation* is the only data structure used in the relational model. A relation is a table with columns and rows. The columns hold the attributes and the rows hold the individual records (tuples).
The basic category relation is defined as follows: R is a relation on given sets D_1, D_2,…, D_n if it is a set of n-tuples, each of which has its first element from D_1, second element from D_2 and so on. The sets D_1, D_2,…, D_n are called the domains of R. The number of the domains is the *degree* of the relation and the number of the tuples in R is its *cardinality* [Zlat1]. Keys establish the relations between the different tables in the relational model. Every attribute or group of attributes, which can uniquely identify every tuple in the relation and does not have a *null* value, can be a key.

SQL

The most widely used relational data model programming language is the Structured Query Language (SQL). The key concept of SQL is that each operation is *mapped* onto the relation. This concept is comparable with searching through a column of a table looking for a matching value or a set of values. SQL uses three key words, SELECT-FROM-WHERE by which a lot of mappings can be performed. By combining different mappings one can also perform the operations union, intersection and difference [ZLAT1].
SQL has a number of impossible operations: identity queries, metadata queries, knowledge queries (explaining the reasoning), qualitative queries (e.g. which road is wider), display of query (the result is always a new table), modifying the database content from the graphics and modifying the graphical presentation. The inadequacy of SQL is currently overcome by utilizing embedded SQL queries [ZLAT1].
Despite all the deficiencies of the relational data model and thanks to the simple concepts and the standard query language, many commercial Database Management Systems (DBMS) have adopted the model that contributed to its successful integration in commercial and administrative applications [ZLAT1].

Object-Oriented (OO) Data Model

The functions of databases have been changing over the years. Nowadays, information systems are not only used for storing numbers and text, but also multi-media, CAD and other complex data structures. Relational databases are not designed to manipulate and store these data forms, whereas object-oriented programming language and object-oriented database management systems are capable of doing so [WAT]
The advantage of the object-oriented approach is the problem-solving strategy encapsulating information, functions and behaviour per object [ZLAT1]. The OO data model is based on the following principles:
Objects and attributes: An object is something in the real world. It can be physical or conceptual. The characteristics of an object are represented by the attributes.
Object instances and classes: Objects of the same type, based on their attributes and behaviour, are called an object class. Every single object in such a class is called an instance of that class.
Generalization/specialization hierarchies: With generalization and specialization, classes can be formed or new object can be specified. Classes can be specializations (subclasses) or generalizations (superclasses) of other classes. For example, the class “Books” can be a specialization of the class “Library”. And visa versa, the class “Library” can be a generalization of the classes “Books” and “CD’s”.
Inheritance: The classes in a generalization/specialization hierarchy inherit information (attributes, behaviour) of their superclasses. In this way, similarities are expressed between objects.
Data abstraction is used to find the most important features of a complicated system. With data abstraction, an abstract model is created to better understand reality.
Encapsulation means that all processing that changes the state of an object is done within that object. An object can only be accessed via its interface; an object cannot directly change any other objects.
Polymorphism means that objects are able to have different forms. This mechanism is often used inside 3D models to speed up real-time navigation.
Association is a mechanism to establish a connection between the objects in two classes. This relationship is not typical for the entire class.
Message passing refers to the communications between objects by sending and receiving messages.

OO modelling and Data modelling use different terminology, but have the same concepts. They both want to represent the real world. Table 1 shows a comparison of the used terminology.

Data Modelling Term	OO Modelling Term
Entity	Object class or classes
Instance	Object
Attribute	Attribute

Table 1: A comparison of data modelling and OO modelling terminology (Watson, 2002)

UML

The Unified Modeling Language (UML) is introduced in 1997 by the Object Management Group (OMG) and has become the common modelling language for object oriented software development. UML is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system (OMG, 2001). The UML architecture is based on a four four-layer metamodel structure. This metamodel has the following layers: user objects, model, metamodel, and meta-metamodel (see table 2). The meta-model is a logical model and defines the language for specifying a model.

Layer	Description	Example
meta-metamodel	The infrastructure for a metamodelling architecture. Defines the language for specifying meta-models.	MetaClass, MetaAttribute, MetaOperation
metamodel	An instance of a meta-metamodel. Defines the language for specifying a model.	Class, Attribute, Operation, Component
model	An instance of a metamodel. Defines a language to describe an information domain.	StockShare, askPrice, sellLimitOrder, StockQuoteServer
user objects (user data)	An instance of a model. Defines a specific information domain.	<Acme_SW_Share_98789>, 654.56, sell_limit_order, <Stock_Quote_Svr_32123>

Table 2: Four Layer Meta-modelling Architecture (OMG, 2001)

Object Oriented vs. Relational

During the last 20 years, relational databases (RDBs) have been established as the most important database technology. Currently, most GISs are implemented either using RDBs or using non-standard application-specific databases [KOF].
When we have a closer look at the nature of GIS data, the disadvantages of RDBs become obvious. GIS data contains arbitrary data types including numeric and short string data, large unstructured data such as textures, complex structured data such as the geometry of buildings and finally compound objects that are comprised of such data [KOF]. The step from 2D to 3D GIS increases both the complexity and the amount of data [KOF].
RDBs lack the mechanisms to deal with this kind of data: Their tabular approach does not allow a suitable modeling of complex hierarchical objects. Although most RDBs support binary large objects (BLOBs) to store graphical data (e.g. geometry, textures), these objects cannot be queried in the same way as other data types [KOF].
The object-oriented data structure helps the systems to work with huge, seamless data sets, to develop object aggregations (composite objects), to solve the multi scale representation of high resolution data, to share objects with other applications running on the desktop, however its most important advantages are, that it fits in with the network based distributed client-server architectures and it supports the creation of standardized geographical object classes [SAR]. The benefit of a RDBs of an easy to use query language (SQL) is still available in a OODBs, but then suited to deal with objects (e.g. UML) [KOF].
Due to the wide spread usage and high technical level of the commercial RDBMS, nowadays the durable storage of object-oriented models is realized in the so-called object-relational database systems (ORDBs) [SAR]. With such a system the compatibility with running RDBs is still available. As the name suggests, ORDBs are actually based on RDBs. As a major advantage they are able to deal with complex data types [KOF].