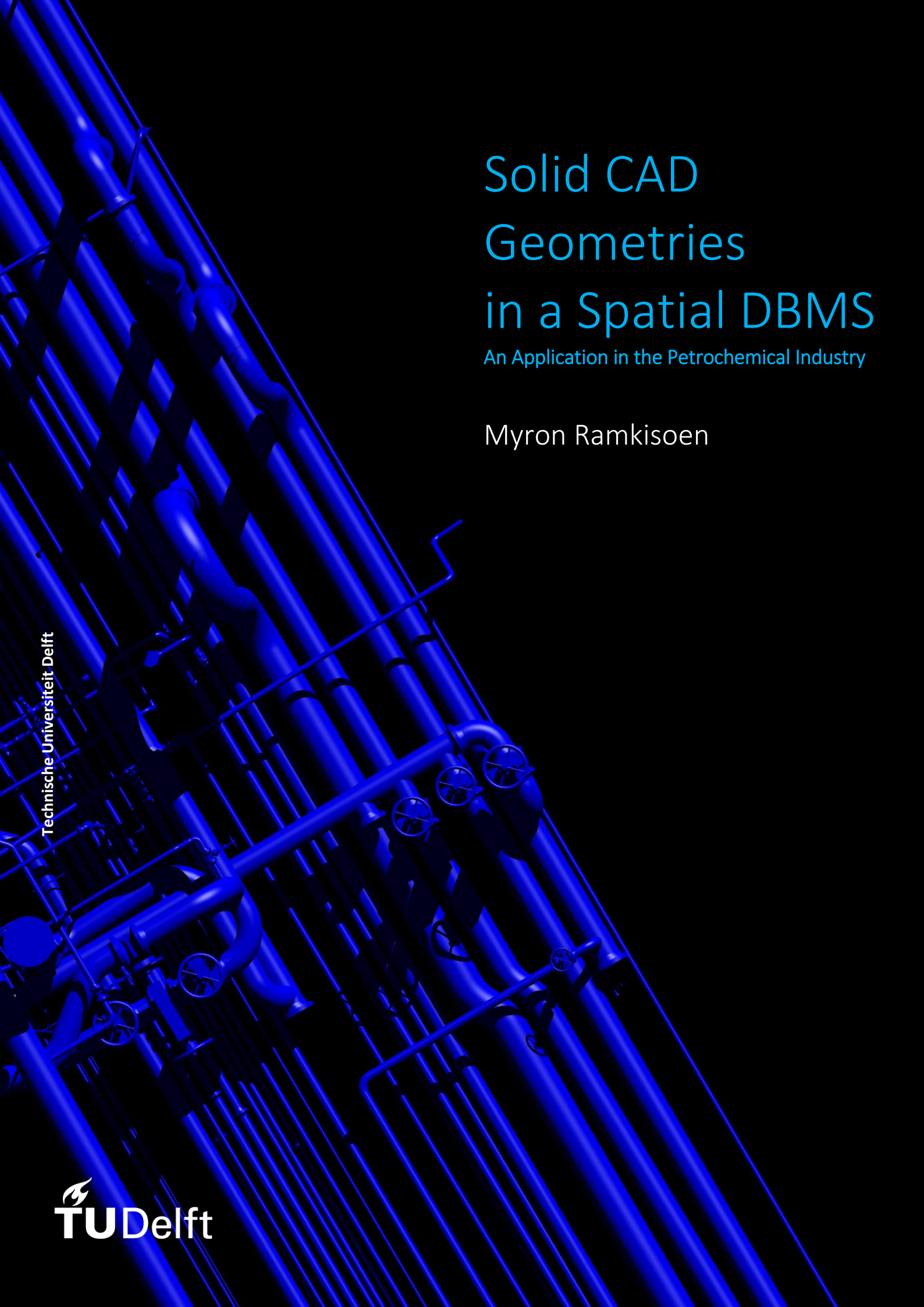


Solid CAD Geometries in a Spatial DBMS

An Application in the Petrochemical Industry

Myron Ramkisoen

Technische Universiteit Delft

TU Delft

Petrochemical data set
Data set courtesy of Fugro GeoServices.
Visualization front page made with Bentley MicroStation,

Solid CAD Geometries in a Spatial DBMS

An Application in the Petrochemical Industry

by

Myron Gianni Widjairadj Ramkisoen

To obtain the Master of Science degree in Geomatics
at the Delft University of Technology,
to be defended publicly on Monday November 2nd, 2015 at 3:15 PM.

Student number:	4078691
Project duration:	April 2015 – October 2015
Thesis committee:	Prof. dr. ir. Peter van Oosterom *TU Delft Graduation Professor*
	Drs. Wilko Quak *TU Delft Daily Supervisor*
	Dr. dipl. ing. S. Zlatanova *TU Delft Third Mentor*
	Ir. Engbert van der Zaag *TU Delft Delegate of Board of Examiners*

This work has been done in cooperation with a graduation internship at Fugro GeoServices, Leidschendam.

Company Supervisor:	Ir. Martin Kodde *Fugro GeoServices Company Supervisor*

TU Delft Delft University of Technology

FUGRO

ABSTRACT

Fugro GeoServices is currently storing their CAD models of the Petrochemical Industry as attributes of 2D lines in a spatial DBMS. This solution works well for web visualization, but not for performing analyses in 3D. The main objective of this graduation project is to create a 3D spatial DBMS with solid CAD geometries as input. The difficulty of this graduation project is the conversion of CAD geometries to 3D GIS geometries. By linking parametric combination models (CAD) with solid polyhedral modelling (GIS) it is possible to reconstruct CAD geometries with help of its parameters as a polyhedron.

This graduation thesis has compared the functionality performance of a 3D parametric DBMS with a 3D non-parametric database. For making an optimal comparison the 3D parametric DBMS had to be optimized as much as possible in terms of query execution time. However a 3D non-parametric DBMS has a better performance, it requires a lot of storage space. The 3D parametric DBMS requires less storage space, but needs some effort for optimizing the performance, such as defining functions for each geometry type, enabling index creation on PostgreSQL's Box3D and making a function that converts this Box3D to a 3D geometry instead of a 2D polygon for querying.

ACKNOWLEDGEMENTS

I would like to thank everyone who has supported me during this graduation project, my supervisors, co-readers, colleagues, family and friends.

At first I would like to thank my supervisors, Peter van Oosterom, Wilko Quak and Martin Kodde, for supporting me during this graduation project. I am very happy I have got this opportunity to explore my knowledge in spatial DBMSs with this graduation topic, which has been provided by Martin Kodde thru Fugro. Martin always encouraged me to explore the most out of myself, and to innovate. After every meeting he always motivated me to come with brilliant solutions.

I am very thankful to Peter and Wilko for approving this graduation topic and having faith in me in graduating in spatial DBMSs. Peter and Wilko always had the time in explaining me things and were very helpful in providing new insights. I will always remember the sociable meetings I had with Wilko.

I would also like to thank Sisi Zlatanova for being my third mentor, my co-reader for this graduation project. Her enthusiastic reaction for being my third mentor made me really happy. I am very thankful for her refreshing view on my graduation project.

Besides my supervisors I would also like to thank my colleagues at Fugro. They were all really helpful for all my questions during my graduation project. In special I would like to thank Bujar Nushi for being one of my co-readers for helping to improve my thesis.

Last but not least, I would like to thank all my family and friends for supporting me during this tough period and getting myself where I am now. In special I would like to thank my parents Eric and Cynthia for always having faith in me.

Thank you all.
Myron

Table of Contents

1

Introduction

This graduation project has been done in cooperation with Fugro GeoServices B.V. in Leidschendam, combined with a graduation internship, to achieve the MSc. Degree in Geomatics. Fugro GeoServices is the provider of the topic of this graduation project and operates in The Netherlands, providing services in the field of geoscience, Geo Information and geo-consultancy. Fugro Geoservices is performing their activities worldwide, focusing on the petrochemical industry, construction, infrastructure, mining and governmental sectors. The topic for this graduation project is related to the petrochemical industry.

The petrochemical industry derives petroleum, with help of steel pipes. These pipes have different shapes, such as cylinders, elbows (bended cylinders) and ending parts (figure 1.1). Petroleum companies hire Fugro to make a 3D model of their industry. This 3D model is created by capturing the industry with a laser scanner. This laser scanner produces a point cloud which is processed to a 3D model in CAD. This 3D model is saved in a database, used for web applications (figure 1.2). Future built models are also stored in the database.

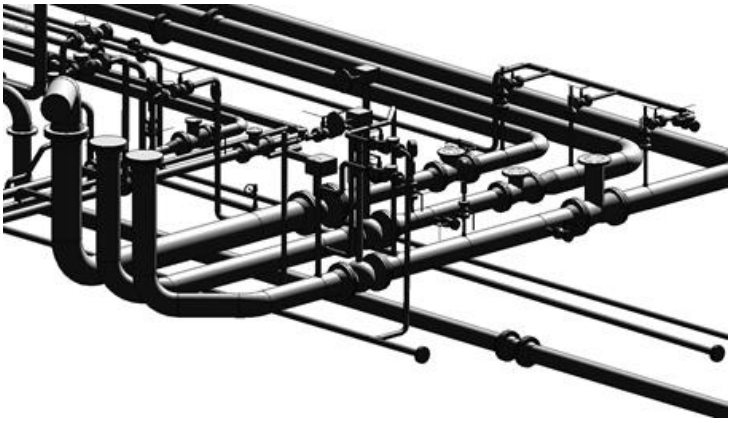


Figure 1.1: Sample geometries of the petrochemical industry.

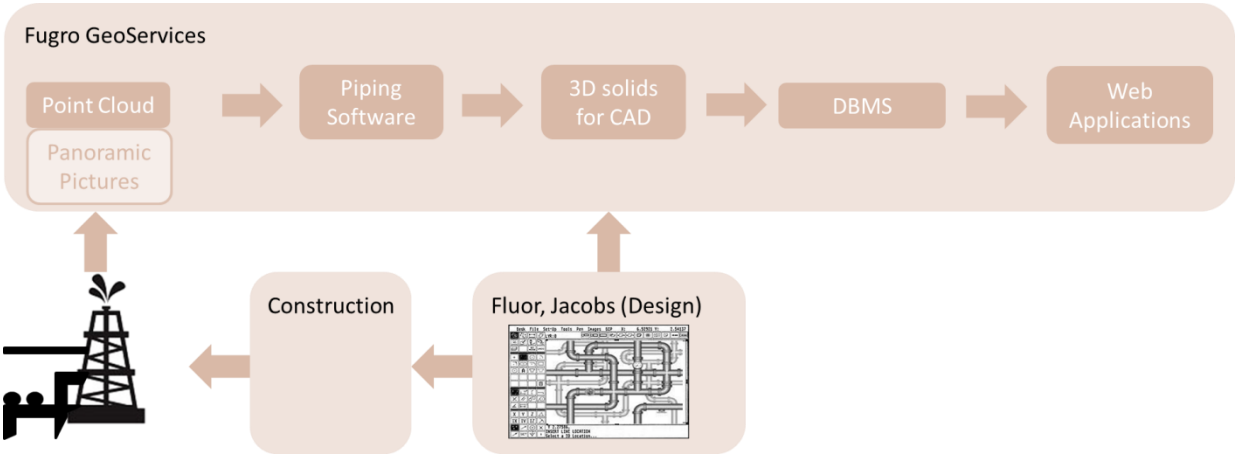


Figure 1.2: Workflow of data processing, from point cloud and CAD files of the petrochemical industry to web applications.

1.1 Problem Statement

The spatial DBMS of Fugro GeoServices saves the geometrical attributes of the 3D CAD data set as attributes of 2D lines, because parametric CAD models are not natively supported by databases. The stored data is therefore only used for web visualization, where the 2D data is parametrically reconstructed in 3D. Fugro GeoServices is exploring how to convert its 2D spatial DBMS to 3D spatial DBMS for performing spatial analyses in 3D. These spatial analyses provide new insights in e.g. disaster management.

For having a transition from a 2D spatial DBMS to a 3D spatial DBMS it is required to reconstruct the geometries in 3D. Research in storing 3D CAD geometries in a 3D spatial DBMS has already been done with an application in the building city environment (Arens, 2003), but not with an application in the petrochemical industry. The difference between both environments is the type of 3D data: the building city environment seldom contains curved geometries. Research on the performance optimization of curved 3D geometries has been done in terms of visualization by Guerrero Iñiguez (2012), but the performance optimization in a DBMS environment has not been done. In the most ideal case the integration of CAD and GIS will lead in having a central DBMS which can be used for CAD design and GIS analyses (figure 1.3) (Pu, 2005). Fugro GeoServices has already made the first step of uniting both worlds: saving the CAD geometries parametrically defined in a DBMS. This graduation project will try to enable the bi-directional flow between DBMSs and GIS analyses.

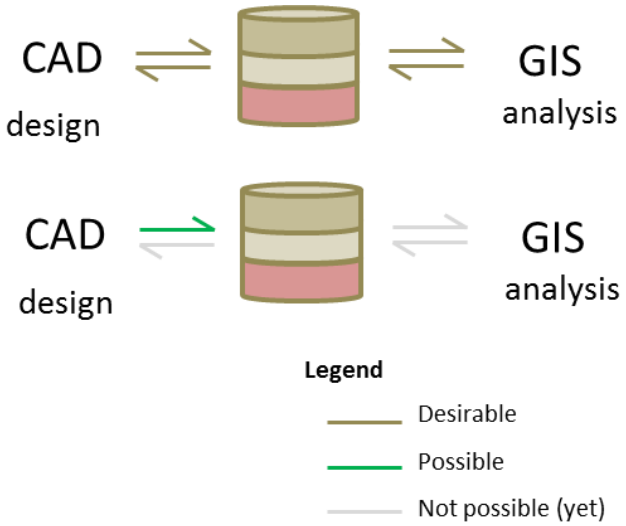


Figure 1.3: Desirable bi-directional flow between DBMSs and CAD & GIS applications (above) versus current one directional flow (below) between CAD applications and DBMSs.

1.2 Objectives

Within this graduation project the storage of CAD solids as a true spatial type in a spatial DBMS will be investigated. The goal is to allow spatial queries on the model data. A part of the work is to demonstrate the visualization of these solids in 3D. The following components are incorporated in this graduation project:

1. Literature study;
2. Investigate existing methods and attempts to store solids;
3. Investigate functional and technical requirements for Fugro's 3D application;
4. Select new storage structures;
5. Analyze performance;
6. Build demonstrator or visualizer;
7. Assessment of solution.

1.2.1 Main Objective

The main objective of this graduation project is to save CAD solids as solid geometries in a prototype DBMS. This prototype system includes a basic visualizer for the validation of 3D spatial queries. After this graduation project Fugro GeoServices will be able to save all its CAD solids of the petrochemical industry in a spatial DBMS with the ability to perform 3D spatial queries. This graduation project investigates which storing method for 3D solid data of the petrochemical industry in a DBMS is the most efficient storing method for Fugro GeoServices. This graduation project will not be focusing on storing the point cloud data of Fugro.

1.2.2 Main Research Question

How is it possible to store and visualize solid geometries in a spatial DBMS suitable for the petrochemical industry?

The emphasis of this graduation project is the storage of 3D CAD solids of the petrochemical industry in a DBMS for selection analysis and not on the visualization. The visualization is basically done for validation purposes. The main goal of this research is having solid CAD geometries stored in a spatial DBMS, with query possibilities.

1.2.3 Sub Research Question

The main research question is divided into the following phases and research questions:

Phase 0: Situation Understanding

1. What is the relationship between CAD and GIS?
2. What are the differences between CAD and GIS?
3. What is a solid geometry?

Phase 1: Data Inventory

4. What are the functional and technical requirements for Fugro's 3D application[1]?
5. What data does the data set of the petrochemical industry consist of?

Phase 2: Potential Exploring

6. Which potential methods exist for the storage of solid CAD geometries in DBMSs?

Phase 3: 3D data storage

7. What are the advantages and disadvantages of the potential methods for storing the solid geometry ?
8. Which method is the most efficient method for storing solid CAD geometries of the petrochemical industry in a spatial DBMS?
9. How is it possible to store the data and incorporate the constrains of Fugro's 3D application?

Phase 4: Performance optimization

10. How can the performance of the 3D DBMS for solid CAD geometries be optimized?

[1] Fugro's 3D application is the usage of the 3D data in practice.

Phase 5: Testing & evaluation

11. How does the 3D DBMS fulfill the requirements of Fugro GeoServices?

Phase A: Visualization

12. How can the queried solids be visualized in 3D?

This graduation project has been conducted with an iterative methodology approach. The iterative methodology approach enables to attack and react on time in an efficient manner. By having an iterative development it is possible to foresee misunderstandings and to encourage active feedback (Kruchten, 2001). This approach ensures the companies satisfaction, because each iteration ensures that the end product fulfills the companies need.

1.3 Relevance to Geomatics

Geomatics is the science of geographical information. The Master Geomatics at the Delft University of Technology is focusing on the vital spatial knowledge of the built environment and incorporates many subjects, including Geo DBMS Management Systems and 3D GeoVisualization. This graduation project focusses on spatial DBMS management and is extended on GeoVisualization within the domains of the master Geomatics.

1.4 Chapters Overview

The structure of this thesis is partially based on Guerrero Iñiguez (2012) and is divided as follow:

- Chapter 2: Background Information

  In Chapter 2 some related work is presented with some background information, related to CAD, GIS, BIM and DBMSs. For the integration of CAD and GIS it is important to dive into both worlds in order to understand the concept of 3D spatial data in CAD and GIS. Besides CAD and GIS it is also important to gain some knowledge of BIM, because BIM is a concept which has emerged out of CAD and GIS. Later, DBMSs are presented, exploring the abilities to have 3D spatial data saved in a DBMS.

- Chapter 3: Preparing the Data Set for a CAD – GIS Environment

  Chapter 3 analyzes the CAD – GIS compatibility with help of the gained knowledge of chapter 2. This CAD – GIS compatibility analysis is used to prepare the data set for a CAD – GIS environment. Chapter 3 also describes the data set of Fugro GeoServices.

- Chapter 4: Representing 3D Solids in a DBMS

  In chapter 4 the 3D solids are reconstructed. This reconstruction leads to a 3D parametric DBMS (with a geometry reconstruction function) and a 3D non-parametric DBMS (saving the geometries as a polyhedron). The performance of both 3D DBMSs (parametric and non-parametric) is compared with each other. This performance comparison has been done in a mini-benchmark setting. The overall purpose of chapter 4 is to optimize the 3D parametric DBMS to a potential DBMS for scaling the data set.

- Chapter 5: Scalability of the Data Set

  In chapter 5 the performance of the optimal 3D parametric DBMS (resulting from chapter 4) is compared to a 3D non-parametric DBMS while enlarging the size of the data set. This performance analysis has been done in a benchmark setting. The data set is equally enlarged in order to see how the performance of the DBMS is affected by the size of the DBMS.

- Chapter 6: Conclusion, Discussion and Recommendations

  Chapter 6 concludes the whole research, based on the results of chapter 5. After the conclusion a discussion will follow, ending with some recommendations.

2

Background Information

This chapter starts by summarizing all related work, related to translating 2D spatial data to 3D spatial data in section 2.1. Section 2.1 also describes the integration problem of CAD and GIS. In order to understand the integration problem of the worlds of CAD and GIS it is important to dive into both concepts, related to 3D modeling in sections 2.2 and 2.3. Both concepts, CAD and GIS, will be compared in order to see the similarities in section 2.4. In section 2.5 the BIM concept will be introduced, which is based on CAD and GIS. Section 2.6 will show some theoretical background regarding DBMSs.

2.1 Related Work

Many studies (Du & Zlatanova, 2006; Guerrero Iñiguez, 2012; Hassan, Ahmad-Nasruddin, Yaakop, & Abdul-Rahman, 2008; Lee & Koh, 2007; Stoter & Salzmann, 2003) have been done for translating 2D data, which represents 3D objects in the real world, to 3D data. This 2D to 3D translation becomes difficult when data of two different environments must be integrated. CAD and GIS are two technologies which are used in civil infrastructural projects. Both technologies are used in different phases (Hijazi, 2011), which has resulted into no unity in standards or interoperability.

CAD and GIS both deal with geometry of the same real world objects, but differ in many aspects, such as ontology, size, storage, analysis, semantics, attributes, world projection, etc. For this reason CAD to GIS conversions are done by hand and cannot be resolved automatically. Both worlds take other aspects into account, because CAD and GIS are used in another phase of the life cycle. CAD focusses more on the shape of the object, than the validness (Kazar, Kothuri, van Oosterom, & Ravada, 2008). According to Van Oosterom, Stoter, and Jansen (2005) there has been much attention for integrating CAD with GIS in the past, but these researches seldom ended

how this could be done, nor specifying the fundamental problems for the integration. CAD software is providing many primitives in its semantics, which are not supported in GIS.

The tendency for uniting the worlds of CAD and GIS is a research which exists for a long time. As can be seen in figure 2.1, this tendency started with comparison researches of CAD and GIS, ending with IFC and CityGML[2] conversions. However the IFC standard is not a real CAD standard, it can be concluded that IFC and CityGML conversions succeeded in uniting both worlds. This conversion succeeded, because the IFC standard is a BIM format. BIM has characteristics of both CAD and GIS (Karimi & Akinci, 2009), which eases the conversion to the other domains (CAD and GIS). Till now there is no unity between CAD and GIS.

2.2 CAD: Computer-Aided Design

CAD, computer-aided design, is a technology which is used to create, modify, analyze and optimize a design with help of computer systems (Groover & Zimmers, 1983). CAD is based upon parametrical design, boundary representation (Bhanu & Ho, 1987; Monedero, 2000) or voxel representation (Kazar et al., 2008) and operates in 2D or 3D in an orthogonal projection of the world (Van Oosterom et al., 2005). The next sections will describe the various methods in more detail.

2.2.1 Parametric Design

Parametric design is a form of 3D modeling by defining a form with help of parameters and relations (Monedero, 2000). Parametric design is done with help of parametric models, where some attributes are fixed and some attributes can vary. These variables are the parameters which can be adjusted by the user. The fixed attributes are the constrains within parametric design. Parametric design allow changes without erasing or redrawing (Hernandez, 2006).

According to Hernandez (2006) parametric design can be divided into three modeling types:

1. **Parametric variations models.** Parametric variation models have been the starting point of parametric design. This parametric model is based on the declarative nature for constructing shapes and allows user modelling according to the desired behavior. This kind of modelling results into a parametrized modeling schema showing the values of the parameters. Changing each value in this parametric schema results into a transformation of the geometry without erasing or redrawing. Changes in the topology[4]

[2] CityGML is a XML-based GIS format and open data model for storing and exchanging 3D city models (OGC, n.d.).
[4] Topology: the number of components and their relations (Hernandez, 2006).

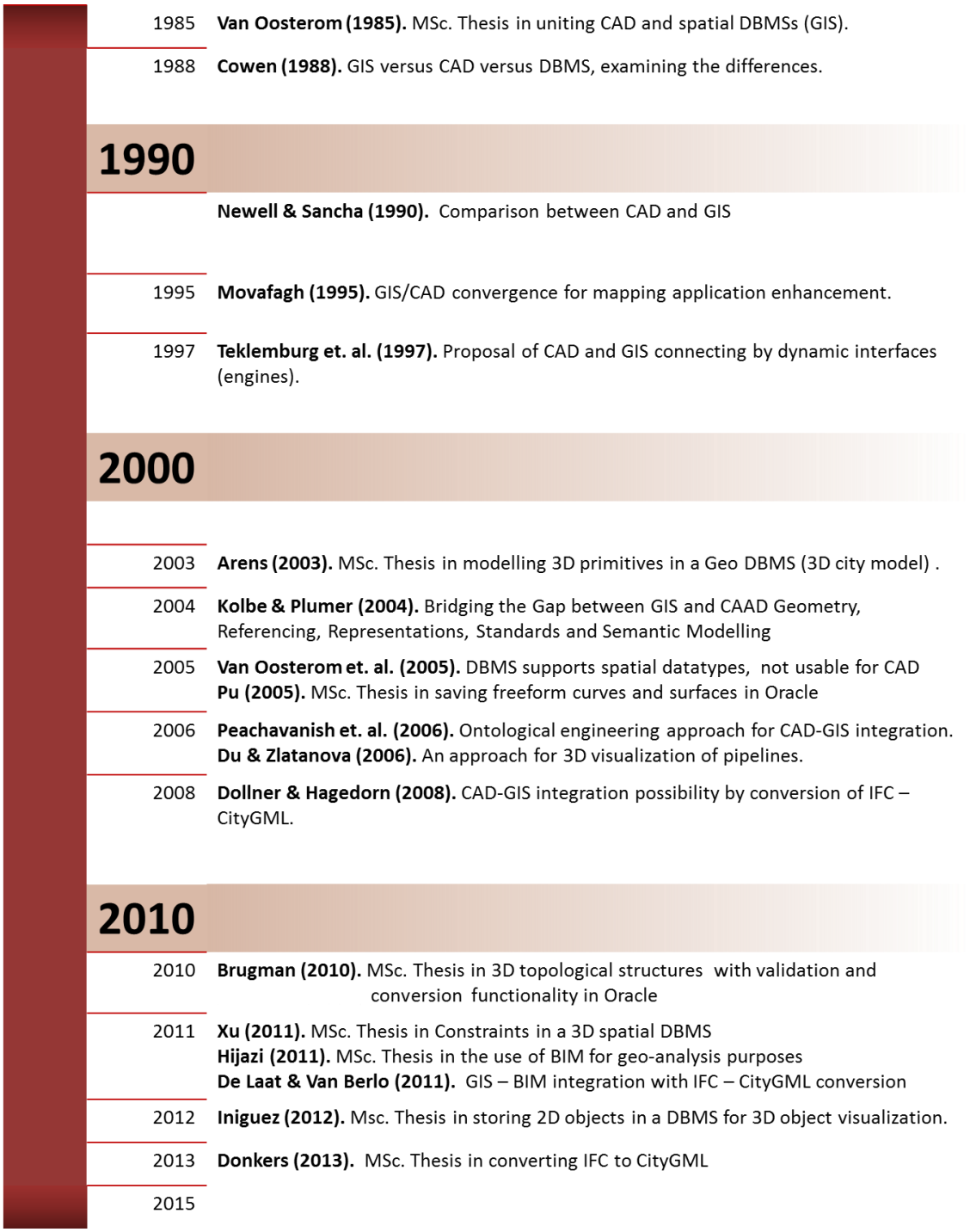


Figure 2.1: Time line with a sample of the past researches for uniting both worlds of CAD and GIS. This time line shows that existing researches started with comparing CAD and GIS and ended with IFC and CityGML conversions.

are not allowed.

This kind of modelling includes NURBS modeling. NURBS, Non Uniform Rational B-Splines, are surfaces based on parametrical curves and are used in computer graphics and CAD to represent model data. The application of NURBS vary from modelling automobile bodies to animated characters (Rogers, 2001). An example of this kind of modelling is Grasshopper-modelling in combination with Rhinoceros 3D (figure 2.2). Rhinoceros 3D is a CAD vendor, based on NURBS.

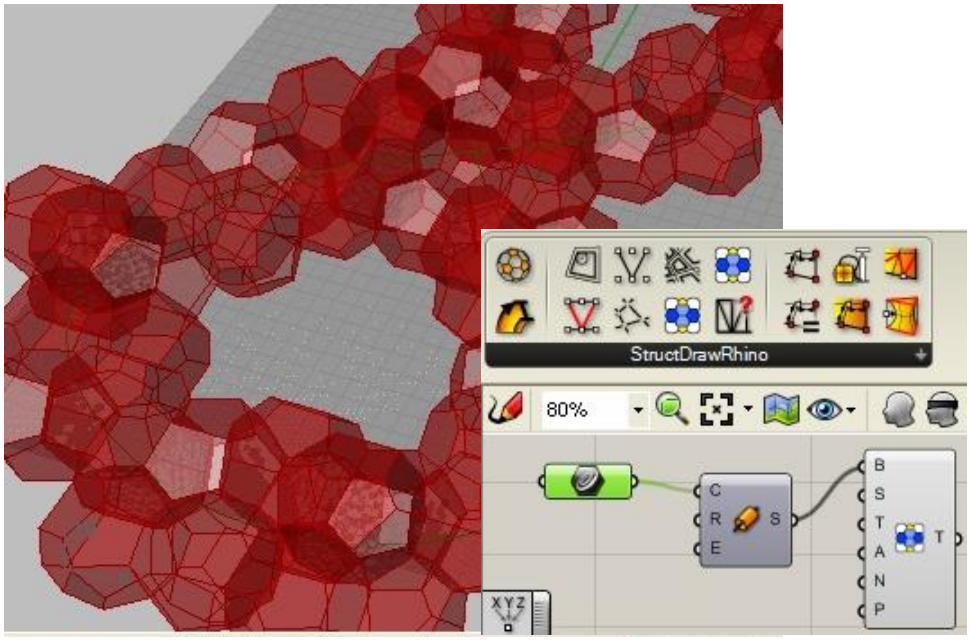


Figure 2.2: Grasshopper modelling: A parametric variation model (Grasshopper, 2010).

2. **Parametric combination models.** Parametric combination models are the most used type of parametric modeling. This kind of modeling is limited to a number of primitive geometrical components, but offers complexity by combining these primitives and the spatial relations between the geometrical components. This allows the creation of new geometrical shapes.

   An example of a parametric combination model is CAD modeling with CSGs, Constructive Solid Geometries. In CSG complicated solids are represented by adding and subtracting primitive solids by Boolean set operators (union, difference and intersection)(Bhanu & Ho, 1987). The CSG tree (structure) is shown in figure 2.3. Figure 2.4 illustrates how CSGs are created using JavaScript in WebGL. The data set of Fugro is modeled with help of CSGs. An example of a primitive is a prismatic volume, having four parameters: location, length, width and height.

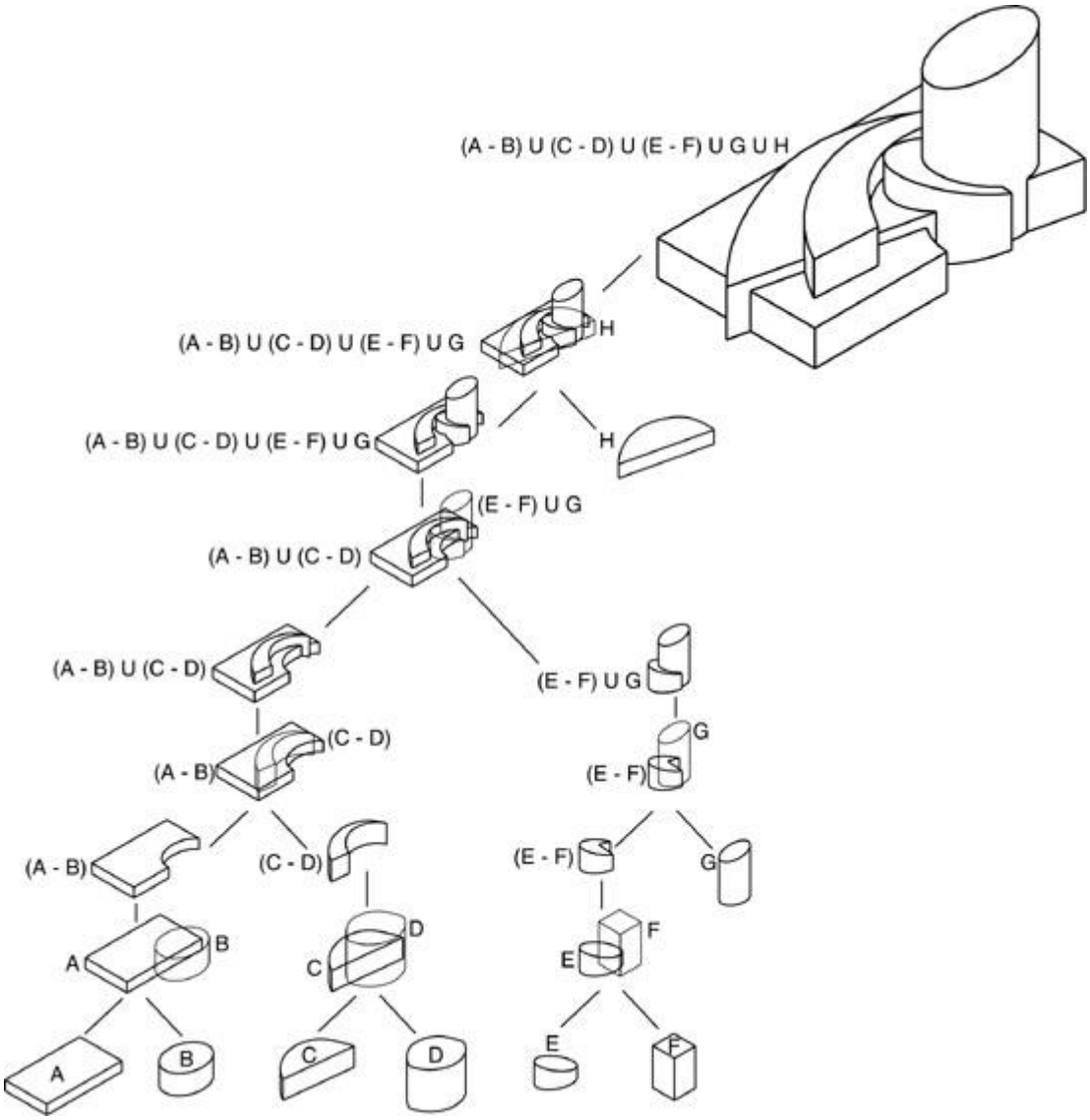
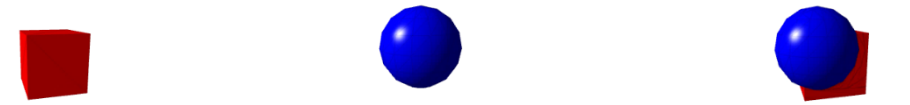


Figure 2.3: Example of a CSG tree structure for creating a 3D model (Qian & Woodbury, 2004).




```
var a = CSG.cube({ center: [-0.25, -0.25, -0.25] });
var b = CSG.sphere({ radius: 1.3, center: [0.25, 0.25, 0.25]
});
a.union(b)
```

Figure 2.4: CSG source code in WebGL with visualization as a parametric combination model (Walles, 2011).

3. **Parametric hybrid models.** Parametric hybrid models are less used than the other two types of parametric modelling. A parametric hybrid model is a combination of both parametric variation- and parametric combination models.

   An example of a parametric hybrid model is Autodesk Revit (figure 2.4). However Autodesk Revit is a BIM, Building Information Model, it is basically a CAD software which allows parametric hybrid modelling. With Autodesk Revit it is possible to reconstruct buildings with help of basic elements, such as walls, roofs, and windows. By adding all basic elements with their topology a new geometry is created (parametric combination model): a house. Each separate basic element in Autodesk Revit is adjustable (parametric variation model). For instance a window can be adapted parametrically by changing certain parameters (e.g. height, width), or even the shape. The topology of the window as a separate element cannot be changed.

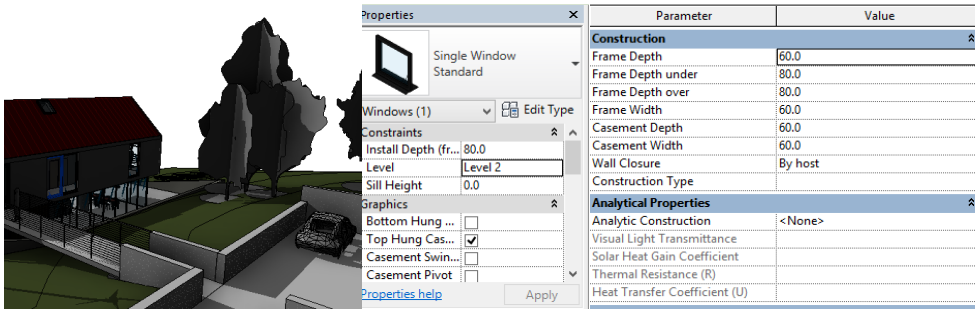


Figure 2.4: Revit parameters of a window. Autodesk Revit as a parametric hybrid model (image has been made with Revit's Sample Project).

2.2.2 Boundary Representation

Boundary representation (figure 2.5) is a skin representation for solid models and form the boundary between model and non-model. The skin consists out of a closed set of surfaces or faces, bounded by a set of edges. These edges are portions of curves, delimited by vertices. At these vertices several faces meet. The data structure can be divided into topology (structure definition of the object) and geometry (form or shape of the object) (figure 2.6) (Stroud, 2006).

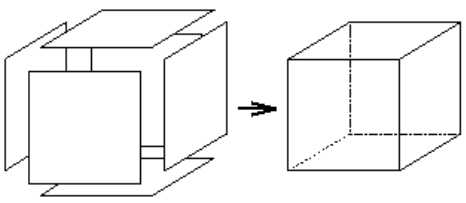


Figure 2.5: Boundary representation (Computer Aided Detector Design, n.d.).

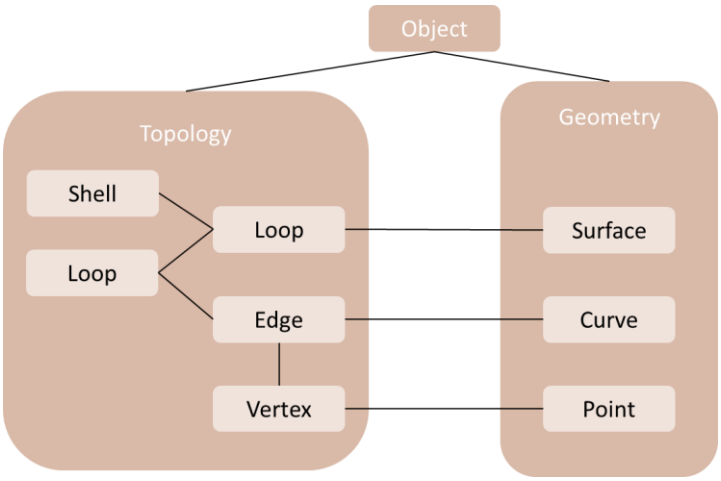


Figure 2.6: Basic data structure boundary representation. Image is based on Stroud (2006).

2.2.3 Voxel Representation

Voxels, also known as volumetric graphics or volumetric imaging, are repeated 3D volume elements in order to model a 3D object in a 3D raster representation. Storing 3D objects as a voxel will result into a rough surface and much storage space (Kaufman, 1989; Wesselingh, 2007). Voxels can be seen as a "Lego block" for describing a 3D object. As can be seen in figure 2.7 the size of the voxel determines the quality of the 3D object: The smaller the voxel, the more accurate the quality of the 3D object will be.

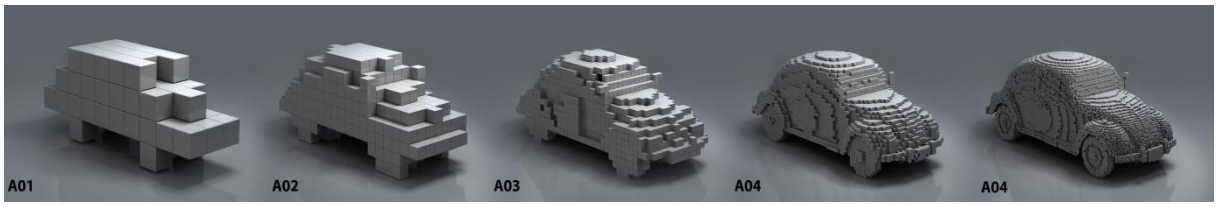


Figure 2.7: Different voxel resolutions, from low quality to high quality (Avin, n.d.).

2.2.4 CAD Vendors

This section gives a short introduction of the CAD vendors used at Fugro GeoServices:

- Autodesk AutoCAD.

  Autodesk AutoCAD allows to model 2D and 3D models, supporting the following file extensions: DWG, DWF[5], DWFx (DWF, based on XML), DGN (Bentley MicroStation format) and PDF (for plotting and drawing). DWF and PDF are used as reference documents in order to protect the original owner's investments, information and integrity in design. DWF and PDF are not that accurate as DWG or DGN (Fane, 2013; Finkelstein, 2014). Autodesk AutoCAD models 3D solids with parametric combination

[5] DWF: Design Web Format. Accurate and compressed vector image representation of drawings.

modelling. These 3D solids can be converted to a boundary solid (Fane, 2013). Another modelling technique that is supported by Autodesk AutoCAD are NURBS surfaces.

- Bentley MicroStation.

  Bentley MicroStation allows to model 2D and 3D models (Pu, 2005). By default Bentley MicroStation uses the DGN file extension. 3D modeling in a DGN file can be done via a primitive solid (parametric combination model) or a SmartSolid (boundary solid representation). It is possible to export this DGN file to a DWG file, but then all primitive solids are automatically transformed to a SmartSolid. In contrary to Autodesk AutoCAD, The DWG file extension in Bentley MicroStation only supports boundary solids.

2.3 GIS: Geographical Information Systems

GIS, geographical information systems, have initially been developed for the storage, retrieval, display of geographical information (Fotheringham & Rogerson, 2013) and analysis, consisting of geographical data, a software package for data processing and a computer system. GIS is aiming for analysis by answering spatial oriented questions (Van Lanen, Bregt, Randen, & Hoosbeek, 1989). One of the main representations of geo-data is vector data. The main types of vector data are points, lines and polygons. Vector data can be stored in both 2D and 3D (Kersting & Döllner, 2002).

In the world of GIS the OGC, The Open GeoSpatial Consortium, desires for standard specifications in GIS in order to support interoperability (Khuan, Abdul-Rahman, & Zlatanova, 2008). The OGC is an international industrial consortium that aims for the participation of companies, government and universities to develop publicly available standards that will support interoperable solutions. The OGC WKT, Well Known Text, is a markup language for representing vector geometry data. WKB, Well Known Binary, saves the same information in binary bites, suitable for storing and transferring the data. Table 2.1 shows the difference of the data structure of one identical point (Wang & Wang, 2010). These coding rules are regulated by the OGC in the Simple Feature Specification (OGC, 1999).

Table 2.1: WKT and WKB example of a point (Wang & Wang, 2010).

OGC Service Specification	Formats
WKT	POINT(1 1)
WKB	0101000000000000000000F03F000000000000F03F

Figure 2.8 shows some examples of WKT data (point, line, polygon (with 1 exterior ring and 0 interior rings) and geometry collection (consisting of 2 points and 1 line)) (OGC, 1999). Note that figure 2.8 shows a 2D WKT data. Figure 2.9 shows an example of a 3D WKT data.

```
POINT (10 10)
LINESTRING (10 10, 20 20, 30 40)
POLYGON ((10 10, 10 20, 20 20, 20 15, 10 10))
GEOMCOLLECTION (POINT (10 10), POINT (30 30), LINESTRING (15
15, 20 20))
```

Figure 2.8: WKT data example of a point, line, polygon and geometry collection (OGC, 1999).

```
LINESTRING (10 10 1, 20 20 1, 30 40 5)
```

Figure 2.9: WKT 3D data example of a line.

Within the world of GIS there are various methods to model 3D data. Figure 2.10 portrays each of these methods: polyhedral modeling (figure 2.10a), NURBS surface modeling (figure 2.10b), voxel modeling (figure 2.10c), octree modeling (figure 2.10d), point clouds (figure 2.10e), TEN modeling (figure 2.10f), boundary representation modeling (figure 2.10g) and 3D Voronoi Diagram modeling (figure 2.10h). The next sections will describe the various methods in more detail.

2.3.1 Polyhedron

A well-known elementary geometry in GIS is the polyhedron as shown in figure 2.10a. In this graduation project the polyhedron definition of Cromwell (1997) will be used:

"A polyhedron is the union of a finite set of polygons such that

i. Any pair of polygons meet only at their sides or corners. […]

ii. It is possible to travel from the interior of any polygon to the interior of any other.

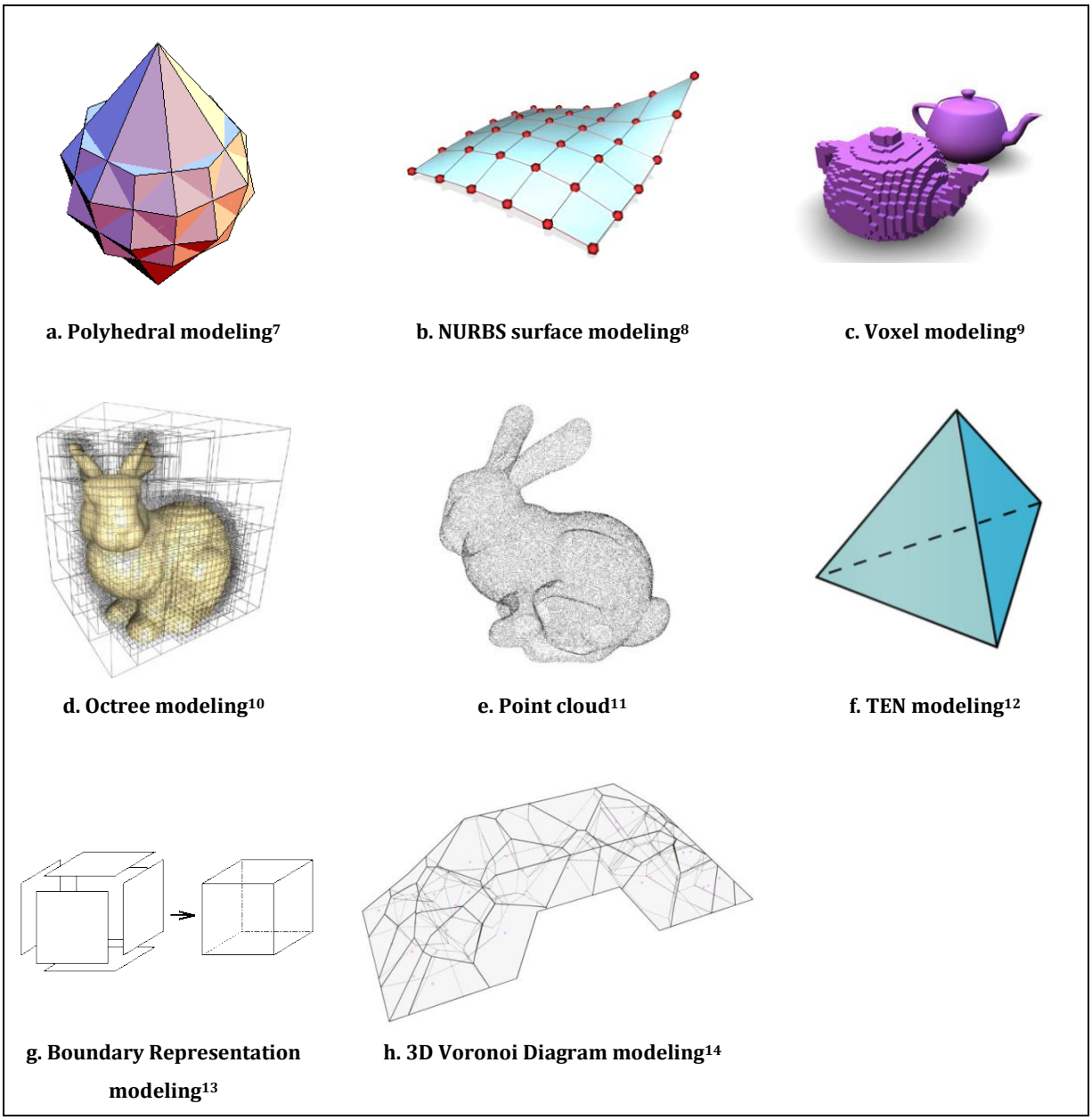
iii. Let V be any vertex and let F_1, F_2, .., F_n the the n polygons which meet at V. It is possible to travel over the polygons F_i from one to any other without passing through V."

iv. An even number of polygons meet at the edge[6].

A polyhedron consists of polygons and defines the boundary of a 3D object. A 3D object can also be a single voxel, a single octree component, a single TEN, a boundary representation or a single

[6] The second point of the polyhedron definition of Cromwell (1997) has been removed and replaced with point iv, because the removed definition of Cromwell (1997) is not valid in all cases.

3D Voronoi component. This 3D object can also be defined with help of NURBS modelling if the NURBS surface is defined as a non-curved polygon.



a. Polyhedral modeling[7] **b. NURBS surface modeling[8]** **c. Voxel modeling[9]**

d. Octree modeling[10] **e. Point cloud[11]** **f. TEN modeling[12]**

g. Boundary Representation modeling[13] **h. 3D Voronoi Diagram modeling[14]**

Figure 2.10: 3D GIS modeling methods **.**

[7] Hart (1997)
[8] 3D-max (n.d.)
[9] Mossman (2013)
[10] Pharr and Fernando (2005)
[11] Mederos, Velho, and De Figueiredo (n.d.)
[12] Ctech (n.d.)
[13] Computer Aided Detector Design (n.d.)
[14] Object-e (2009)

2.3.2 NURBS

NURBS, Non-Uniform Rational B-Spline, (figure 2.10b) are a mathematical representation used for free-form modeling of surfaces. NURBS allow more freeform by parametric modeling with weight, using control points (Pu, 2005; Rogers, 2001). NURBS surfaces are parametrically modeled, allowing analytic and freeform shape representation and are also known as a parametric variation model (CAD).

2.3.3 Voxels

Besides CAD, voxels (figure 2.10c) are also used in GIS for 3D representation (section 2.2.3). Voxels are the 3D counterpart of the 2D pixels, which can represent a material, color, texture, translucency ratio, or other characteristics per voxel (Wesselingh, 2007).

2.3.4 Octrees

In solid modeling octrees (figure 2.10d) are a sub method of the cell decomposition methods. The cell decomposition method divides the object space into unit-sized elements, cubes or spheres, to represent shapes as a collection of these elements. It is not possible to divide the object space into infinitely small elements due computer limitations. The cell decomposition method is also known as voxel modelling. A refinement of this technique is octree modeling. With octrees the bounding box of a 3D element is computed which is called the 3D cell. This 3D cell is partitioned into smaller 3D cells until the desired resolution is computed: a 3D mesh. Every partitioning divides the cell into eight children cells. With computations each children cell is classified as filled, partially filled or empty. The partially filled children cells are subdivided until the desired resolution is achieved (Peng & Kuo, 2005; Stroud, 2006). The recursive subdivision into eight equal cells is shown in figure 2.11.

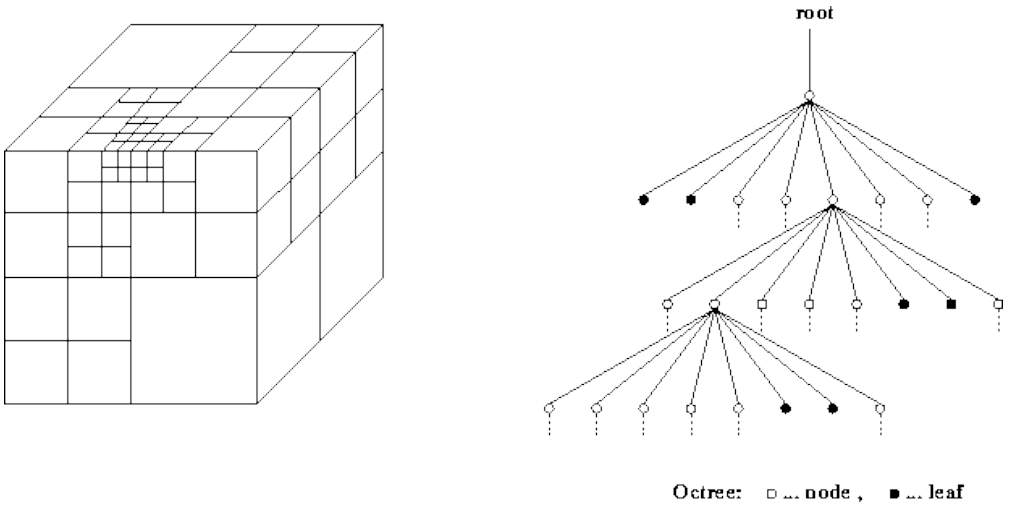


Figure 2.11: Recursive subdivision into 8 equal cells for octree modelling (Stippel, 1993).

Both octrees and voxels involve modelling with a 3D cubical element. The main difference between both is that voxels allow modeling with one single defined element and octrees allow partition of the 3D element, what results into smaller elements. This makes octrees more efficient in retrieving a higher resolution. Voxels have to adjust all elements to smaller elements, but octrees only have to adjust the element at necessary places.

2.3.5 Point Cloud

A point cloud (figure 2.10e) consists of a big amount of points. The amount of points depends on the density of the point cloud. Before modeling a 3D object it is evident to filter out the noise, usually done with a smoothing filter. For 3D modeling this filter will remove all the sharp edges, but these sharp edges can be reconstructed with an algorithm (Mitra, Nguyen, & Guibas, 2004).

A point cloud does not represent a volume, but points. It might be possible to define a boundary volume with help of voxels, by positioning voxels at the position of a point. With help of interpolation the empty spaces between the points or voxels can be filled, so a boundary representation arises consisting of voxels.

2.3.6 TEN

According to Penninga and Van Oosterom (2008) "a TEN is a simplicial complex consisting only of face-connected 3-simplexes that model the full 3D domain." A TEN (figure 2.10f) is a 3D variant of a TIN, Tetrahedral Irregular Network, and is the simplest polyhedron possible: a pyramid with a triangular ground surface. This tetrahedron consists of 4 points. The only restriction of these four points is that all the points must not lay on one plane. Multiple tetrahedra can model a 3D model, as shown in figure 2.12 (Wesselingh, 2007).

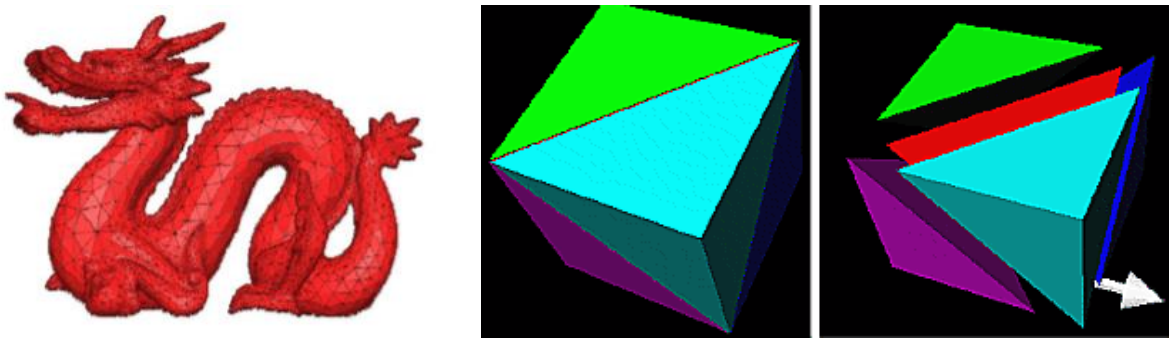


Figure 2.12: A network of tetrahedral, modeling a 3D model (Eckel, n.d.; Wesselingh, 2007).

Relating a TEN to a point cloud it is possible to reconstruct a solid geometry out of a point cloud, if the solid geometry is convex. By triangulating the point cloud and reconstructing these triangles as such that a tetrahedron exists, a solid geometry can be constructed. If it is not

possible to reconstruct these triangles as a tetrahedron a boundary representation can be created by reconstructing triangles with only 3 points or vertices. This triangle reconstruction with 3 points or vertices is also known as the Delaunay triangulation and can also help in reconstructing voxels with help of boundary boxes, which has been implemented by Sisi Zlatanova and Pirouz Nourian for MonetDB, based on Laine (2013). TENs are also related to voxels, because both methods are subdividing space into units.

2.3.7 Boundary Representation

Boundary representation is a collection of connected surfaces which together represent a 3D solid model (Wesselingh, 2007). "A boundary representation [...] specifies the location of the vertices, their connectivity, and a description of how they should be interpolated or approximated by a simple surface (such as a polyhedron [...] or parametric patch)" (Rossignac, 2002). Boundary representation is also used by CAD for 3D modelling.

Both boundary representation and NURBS represent the boundary between object and non-object. The main difference between both methods is the parametrical character of the boundary when modeling a NURBS surface. A boundary representation is a flat surface or one dimensional curved surface (according to the 1st order polynomial) between vertices, but NURBS allow parametrically curved surfaces (of higher order polynomial). This allows more flexibility to NURBS than boundary representations.

2.3.8 3D Voronoi Diagram

The basic principle of both the 2D and 3D Voronoi Diagram is based on the distance function between points (Hoff III, Keyser, Lin, Manocha, & Culver, 1999). The 3D Voronoi Diagram is based on the assumption that a point is seen as a centroid, which is the center of a mass with respect to a given density function, of the corresponding Voronoi region (Q. Du, Emelianenko, & Ju, 2006).

Relating the 3D Voronoi Diagram to a TEN both methods are duals (figure 2.13) and are using points for reconstructing a surface. A TEN uses the points as boundary points for triangulation and the 3D Voronoi Diagram uses the points as the center of the reconstructed geometry. The 3D Voronoi Diagram is also related to voxels. Both methods are subdividing the space into units.

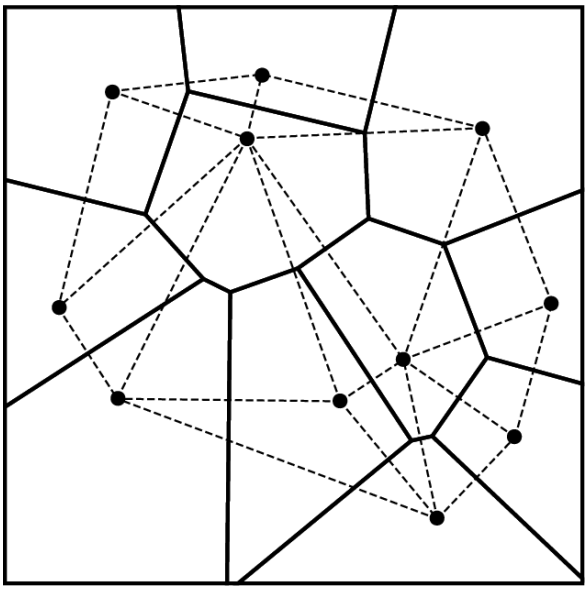


Figure 2.13: Relating a TIN (2D TEN)(dashed line) to a 2D Voronoi Diagram (continuous line)(Kruip, Paardekooper, Clauwens, & Icke, 2010).

2.4 CAD versus GIS

Both CAD and GIS are founded in the early computer graphics. CAD applications are focusing on design and often lack a robust attribute model. GIS applications are focusing spatial analysis and are relying on a DBMS attribute model (Karimi & Akinci, 2009). However CAD and GIS have been developed separately both worlds meet at some places. The similarities of CAD and GIS, based on section 2.3, are shown in table 2.2.

Table 2.2: Modeling method comparison, relating GIS to CAD.

GIS	**CAD**				
	Parametric design			Boundary representation	Voxels
	Parametric variant modeling	Parametric combination modeling	Parametric hybrid modeling		
Polyhedron	○	○	○	●	○
NURBS	●	○	○	●	○
Voxels	○	○	○	○	●
Octrees	○	○	○	○	●
Point cloud	○	○	○	○	○
TEN	○	○	○	○	●
Boundary Rep.	○	○	○	●	○
3D Voronoi Diagram	○	○	○	○	●

Table 2.2 illustrates that the CAD modeling techniques meet the GIS modeling techniques with parametric variant modeling, boundary representations and voxels. The parametric variant modeling uses NURBS as modeling technique, which is also used as a GIS modeling technique. Both CAD and GIS use boundary representations (which is similar to a polyhedron and 3D modeling with NURBS) and voxels (octrees, TEN and 3D Voronoi Diagram are also regarded as a subclass of voxels).

An important term within this graduation project is the term solid geometry. Both worlds, CAD and GIS, have another perception of a 3D solid. Natively a solid geometry was defined as a geometry in the three dimensional Euclidean space. In the sense of computer modeling there are mainly four representation techniques for solid modelling (Stroud, 2006):

1. **Cell decomposition.** Dividing the object space into unit-sized elements.
2. **General sweeping.** Extruding 2D shapes along general curves.
3. **Set theoretic.** This technique for solid modeling involves two kinds of solid representation:
    i. Representing solids as a combination of primitive shapes with Boolean operations.
    ii. Representing solids as a collection of object boundaries, which allow intersections, based on Boolean operations.
4. **Boundary representation.** Representing the boundary between object and non-object.

With the separate development of CAD and GIS both worlds have been using other semantical interpretations for solid modelling. The leading definition in the world of GIS for solid geometry is the definition of the OGC: a boundary surface. The OGC defines a 3D solid object as a GM_Solid, which is a subclass of GM_Primitive (Khuan et al., 2008). This notion complies with the fourth solid modelling technique: boundary representation. However GIS also uses the first solid modeling technique (octrees, voxels, TEN and 3D Voronoi Diagram). As written in section 2.2 CAD refers to a parametric modelling, boundary representation or voxels for solid modelling. This notion complies with all solid modeling techniques: parametric design for the second and third modelling technique, boundary representation for the second, third and fourth solid modelling technique and voxel representation for the first solid modeling technique. Table 2.3 summarizes the division of all CAD and GIS methods according to the solid modelling representation techniques. Note: TENs are using a similar solid representation technique as a cell decomposition.

Table 2.3: Division of solid modelling representation techniques of the world of CAD and GIS.

	Solid modelling			
	Cell decomposition	General sweeping	Set theoretic	Boundary representation
GIS	●	○	○	●
CAD	●	●	●	●
GIS				
Polyhedron	○	○	○	●
NURBS	○	○	○	●
Voxels	●	○	○	○
Octrees	●	○	○	○
Point cloud	○	○	○	○
TEN	●	○	○	○
Boundary Rep.	○	○	○	●
3D Voronoi Diagram	●	○	○	○
CAD				
Parametric design	○	●	●	○
Boundary represent.	○	●	●	●
Voxels	●	○	○	○

Note that set theoretic and boundary representation both involve boundary representations. The difference between both is only in the modelling technique. Both CAD and GIS share boundary representation (Mesh versus a boundary representation or polyhedron) as a similar modelling technique.

For this graduation project it is evident to make a distinction between two kinds of solid modelling:

- **True solid:** A true solid is defined as a solid with volume.
- **Boundary solid:** A boundary solid is defined as a boundary surface representing the boundary between object and non-object, with a missing volume concept.

2.5 BIM: Building Information Model

In the field of architectural and civil engineering CAD has had a steady evolution, which has resulted into BIM: Building Information Model. BIM includes some intelligent behavior of GIS, such as associated data and rules, non-redundant geometry (topology), automatic modification

of associated geometries, and others. BIM is a 3D design tool with parametric objects that behave or interact with other objects. With a BIM software it is possible for vendors to provide objects, which differ from the standard objects from the BIM software. BIM allows data sharing with other project members and serves as a design communication medium. BIM is collaboratively developed with all stakeholders (input) (Karimi & Akinci, 2009).

IFC, Industrial Foundation Classes, is the international standard data format to describe exchange and share information within the building and facility management industry sector. IFC is an open standard developed for BIM by the international organization buildingSMART and is based on the EXPRESS language as part of the STEP (Standard for the Exchange of Product model data) standard[15], which enables to define IFC models using XML (Hijazi, 2011).

According to Hijazi (2011) IFC include the following key contents:

- Object-oriented and semantic model
- Re-use of building information through the whole building lifecycle.
- 3D representation of building models
- Spatially data model (spatial relationships between building elements is maintained hierarchically)

Besides the notion raised by Hijazi (2011) IFC also includes more domain semantics into the BIM model. IFC is a shared data model (figure 2.14), and has been accepted by the Dutch 'Forum en College Standaardisatie' as an open standard. This obligates governmental organizations to use the IFC standard when working with BIM (Donkers, 2013).

Besides creating parametric combination models and boundary representation models, the IFC standard also includes swept solids and half-space solids. Swept solids are solids created by revolution and linear extrusion of a solid, which are obtained by sweeping or extruding a planar face (with may contain holes). Half space solids (figure 2.15) are semi-infinite solids (on one side of a surface) which can be limited by a box domain (BuildingSMART, n.d.-b). IFC is parametric design, because the IFC standard for BIM is based on CAD. Looking at e.g. a sphere, the IFC standard saves the following attributes for constructing this sphere: center and radius (BuildingSMART, n.d.-c). IFC supports boundary representations, NURBS and parametric combination models (CSG) (BuildingSMART, n.d.-a).

[15] ISO 103030 (Hijazi, 2011).

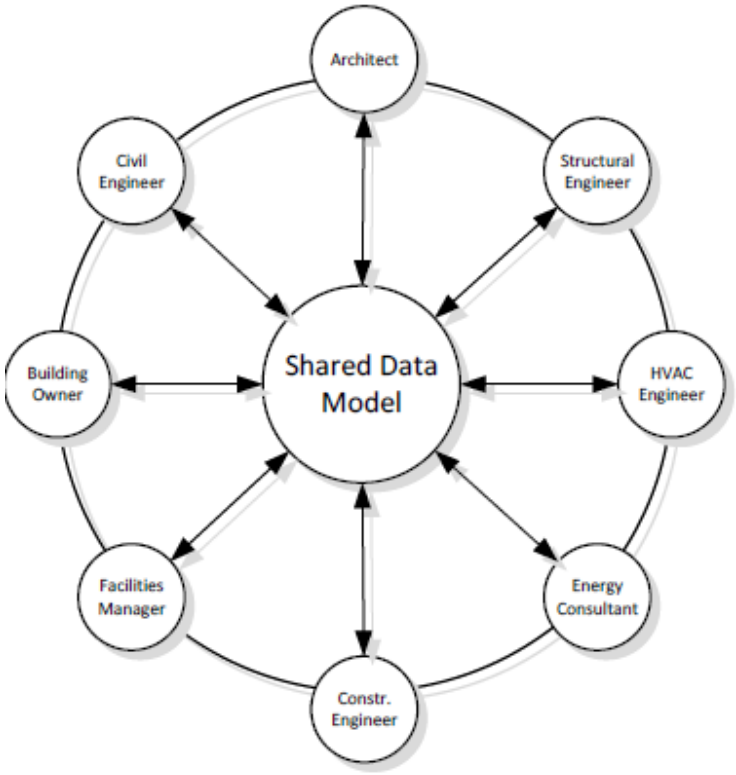


Figure 2.14: IFC as a shared data model (Donkers, 2013).

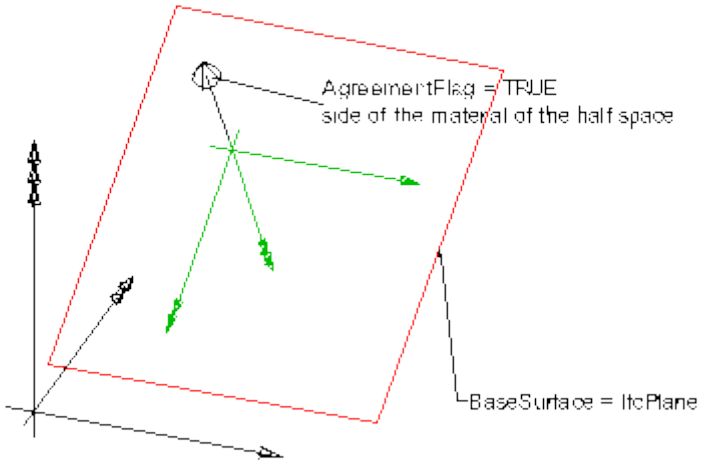


Figure 2.15: IFC half space solids example (BuildingSMART, n.d.-b).

Recently much research has been done for uniting the worlds of CAD and GIS with help of the IFC standard (Benner, Geiger, & Leinemann, 2005; Donkers, 2013; I-Chen & Shang-Hsien, 2007; Nagel, Stadler, & Kolbe, 2009). However these researches were mainly focusing on the transformation of architectural elements, such as walls, spaces and doors into CityGML and GML (Hijazi, 2011).

2.6 DBMS: Database Management System

A DBMS, Database Management System, is a software which handles all access to the database (Date, 1975) and controls the storage & retrieval, addition/deletion, data definition and journaling of data, which consists out of a kernel code (for managing memory and storage), repository data (data dictionary) and a query language (enables to access the data) (Ashdown & Kyte, 2014; Chorafas, 1983). Figure 2.16 visualizes the four parts of a DBMS. In order to access and manage the data a DBMS application is needed (Ashdown & Kyte, 2014).

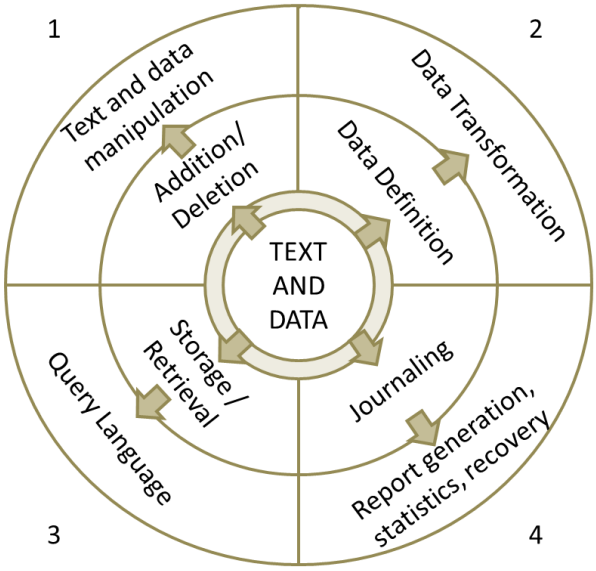


Figure 2.16: The four parts which a DBMS involves (Image is based on Chorafas (1983)).

A DBMS has traditionally been used to handle large volumes of data and ensures consistency and integrity of data. The development of the DBMS went from managing administrative data to managing spatial data. Spatial data used to be organized by having the administrative data in a Relational DBMS (RDBMS) separately from the spatial data in a GIS (Zlatanova, 2006). A RDBMS has the relational model (structure, operations and integrity rules) as basis and move, store and retrieve data from or into a DBMS, so it can be manipulated by applications. A RDBMS contains logical operations[16] and physical operations[17] (Ashdown & Kyte, 2014). The separate spatial data was managed in a separate single file in a proprietary format, linked with common object identifiers (Vijlbrief & van Oosterom, 1992). Now the dual architecture is replaced by a layered architecture and spatial data is supported by DBMSs (Zlatanova, 2006), it is possible to manage spatial data into a single spatial DBMS.

[16] Logical operations specifies what content is required (Ashdown & Kyte, 2014).
[17] Physical operations specifies how things should be done (Ashdown & Kyte, 2014).

One of the first approaches of combining a spatial DBMS with a GIS front-end has been the GEO++ system, by having a DBMS/GIS architecture for querying spatial data. This system is regarded as a complete GIS, based on a spatial DBMS (Van Oosterom, Vertegaal, Van Hekken, & Vijlbrief, 1994; Vijlbrief & van Oosterom, 1992).

2.6.1 Spatial DBMS

A spatial DBMS enables to maintain spatial data types (such as point, lines, polygons) in contrary to traditional DBMSs. With a spatial DBMS it is possible to perform functions on spatial data types, like returning geometric information, performing basic geometric transformation, maintaining valid geometry, etc. (Pu, 2005). There are many DBMS which support spatial data. One of the biggest DBMSs with spatial support are Oracle, PostgreSQL, MonetDB, Microsoft SQL server and MySQL.

Oracle

Oracle is a RDBMS that implements object-oriented features (user-defined types, inheritance, polymorphism), which is called an object-relational database management system (ORDBMS). An ORDBMS makes it possible to store complex business models in Oracle DBMS (Ashdown & Kyte, 2014). The spatial extension of Oracle DBMS is called Oracle Spatial. Oracle Spatial enables the storage, retrieval, update and query of spatial features in an Oracle DBMS using SQL schema and functions. 3D spatial data is supported by Oracle Spatial (Ashdown & Kyte, 2014).

PostgreSQL

PostgreSQL is an open source ORDBMS, which supports the SQL standard to a large extend. PostgreSQL can be extended by the user by adding new data types, functions, operators, aggregate functions, index methods and procedural languages (PostgreSQL, n.d.). PostgreSQL supports spatial data with the PostGIS extension, which allows PostgreSQL to be used as a back-end spatial DBMS for GIS applications. PostGIS follows the OpenGIS Simple Feature Specification for SQL and supports 3D spatial data (The PostGIS Development Group, n.d.).

A new development for 3D operations is the SFCGAL extension of PostGIS. SFCGAL is a C++ wrapper around CGAL for advanced 2D and 3D functions (The PostGIS Development Group, n.d.). SFCGAL includes a solid geometry type and is based on the ISO 19107 Spatial schema of the OGC (SFCGAL, 2013).

MonetDB

MonetDB is an open source column-oriented DBMS designed for multi-core parallel executions on desktops for reducing complex query processing. MonetDB supports many programming interfaces, such as JDBS, ODBC, PHP, Python, RoR,C/C++ and Perl (MonetDB, n.d.-a) and stores data in a binary structure, which are called a BAT (Binary Association Table). BAT represents a

mapping from an OID[20] to a base type value (Goncalves & Kersten, 2011). MonetDB/SQL has an interface which supports 2D spatial data (OGC compliant), but has no support for 3D spatial data (MonetDB, n.d.-c). MonetDB will be supporting 3D data (point clouds and voxels) in the future (MonetDB, n.d.-b).

Microsoft SQL Server
Microsoft SQL server is a RDBMS, based on SQL with XML support for web applications (Microsoft, n.d.-c). Microsoft SQL server 2014 supports spatial data in a Euclidian (flat) coordinate system and in a round-earth coordinate system. Microsoft SQL server only supports 2D data (Microsoft, n.d.-d).

MySQL
MySQL is an open source RDBMS, which is based on SQL and developed, distributed and supported by Oracle Corporation. The source code of MySQL is available to study and to adapt according to the user's preferences. With MySQL it is possible to access DBMSs via the Internet. MySQL supports spatial data only in 2D (MySQL, 2015).

2.6.2 Spatial Queries
A spatial DBMS can perform spatial queries on spatial data. 2D spatial queries are based on the 2D topological relationships as shown in figure 2.17.

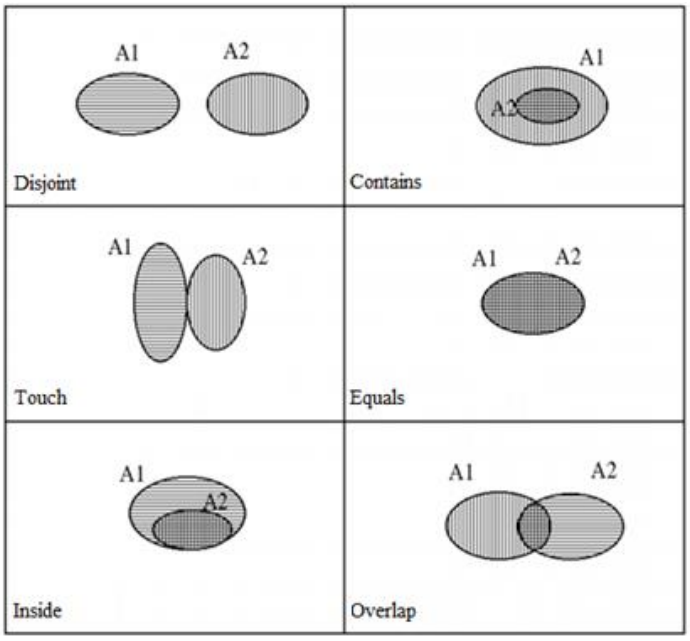


Figure 2.17: Visualization of six different 2D topological relationships (based on Clementini et al. (1993)).

[20] OID: object identifier. Unique value in a database of an object (Ramakrishnnan, 2003).

Figure 2.17 shows the following topological relationships: disjoint, contains, touch, equals, inside and overlap (Clementini, Di Felice, & Van Oosterom, 1993). In order to perform spatial queries in 3D the topological relationships of figure 2.17 have to be converted to 3D topological relationships. The conversion of 2D topological relationships to 3D topological relationships is called the volume and volume relationship, which has been proposed by Egenhofer (1995). With the conversion of the 2D topological relationships to the 3D topological relationships two additional relationships are proposed: covers and cover by. Figure 2.18 shows an overview of all possible volume and volume relationships in relation to figure 2.17 Egenhofer (1995).

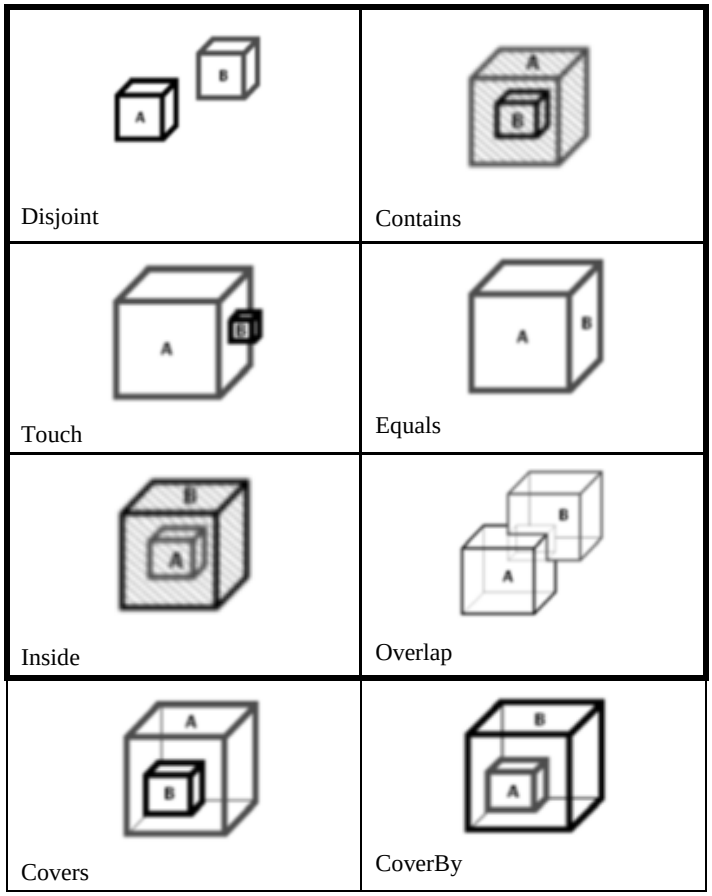


Figure 2.18: Visualization of the proposed volume and volume relationships by Egenhofer (1995), highlighting figure 2.17 (Image based on Zlatanova (2000)).

The distinction between all the volume and volume relationships has been made with help of the 9-IM. The 9-IM (9-intersection model) is a 3x3 matrix as shown in Eq. 2.1 (Egenhofer, 1995).

$$\widetilde{\varsigma_9}(A,B) = \begin{pmatrix} A \cap B & A \cap \delta B & A \cap B^- \\ \delta A \cap B & \delta A \cap \delta B & \delta A \cap B^- \\ A^- \cap B & A^- \cap \delta B & A^- \cap B^- \end{pmatrix} \quad (2.1)$$

where:

A = interior of volume A

B = interior of volume B

δA = boundary of volume A

δB = boundary of volume B

A^- = exterior of volume A

B^- = exterior of volume B

The 9IM or Egenhofer- Matrix is an extension of the 4IM (Four Intersection Model) (Strobl, 2008) and incorporates six object parts of two volumes: interior, boundary and exterior (Eq. 2.1). These six object parts show a topological relation of two volumes. In figure 2.19 the volume and volume relationships of figure 2.18 are shown with the corresponding 9IM. The 9IM is a binary topological relationship model, so true is indicated with 1 and false is indicated with 0 (Egenhofer, 1995).

$\begin{matrix} & B & \partial B & B^- \\ A & 0 & 0 & 1 \\ \partial A & 0 & 0 & 1 \\ A^- & 1 & 1 & 1 \end{matrix}$ Disjoint	$\begin{matrix} & B & \partial B & B^- \\ A & 1 & 1 & 1 \\ \partial A & 0 & 0 & 1 \\ A^- & 0 & 0 & 1 \end{matrix}$ Contains
$\begin{matrix} & B & \partial B & B^- \\ A & 0 & 0 & 1 \\ \partial A & 0 & 1 & 1 \\ A^- & 1 & 1 & 1 \end{matrix}$ Meet	$\begin{matrix} & B & \partial B & B^- \\ A & 1 & 0 & 0 \\ \partial A & 0 & 1 & 0 \\ A^- & 0 & 0 & 1 \end{matrix}$ Equals
$\begin{matrix} & B & \partial B & B^- \\ A & 1 & 1 & 1 \\ \partial A & 0 & 1 & 1 \\ A^- & 0 & 0 & 1 \end{matrix}$ Inside	$\begin{matrix} & B & \partial B & B^- \\ A & 1 & 1 & 1 \\ \partial A & 1 & 1 & 1 \\ A^- & 1 & 1 & 1 \end{matrix}$ Overlap
$\begin{matrix} & B & \partial B & B^- \\ A & 1 & 0 & 0 \\ \partial A & 1 & 1 & 0 \\ A^- & 1 & 1 & 1 \end{matrix}$ Covers	$\begin{matrix} & B & \partial B & B^- \\ A & 1 & 0 & 0 \\ \partial A & 1 & 0 & 0 \\ A^- & 1 & 1 & 1 \end{matrix}$ CoverBy

Figure 2.19: The volume and volume relationships with their 9IM (Egenhofer, 1995).

2.6.3 Data Exchange Formats

"Data exchange is the problem of taking data structured under a source schema and creating an instance of a target schema that reflects the source data as accurately as possible" (Fagin, Kolaitis, Miller, & Popa, 2005). Data exchange is used for data transfer between existing, independently created applications (Fagin et al., 2005). The most used languages for data exchange are RDF[21], XML[22], JSON[23], YAML[24], Gellish[25] and CSV[26]. These languages are compared

[21] RDF, Resource Description Framework, is a framework for the representation of information on the web (Klyne & Carroll, 2006).

with the biggest DBMSs from section 2.6.1 in table 2.4. Table 2.4 gives an overview which DBMS is compatible with which data exchange formats. This table only incorporates existing extensions.

Table 2.4: DBMS format ready comparison for data exchange[27].

	RDF*	XML	JSON	YAML	Gellish**	CSV
Oracle	●	●	●	○	◍	●
PostgreSQL	◍	●	●	●	◍	●
MonetDB	●	●	●	○	◍	●
MS SQL	●	●	●	○	◍	●
MySQL	◍	●	●	○	◍	●

* RDF script of W3C (W3C, 2003) works with all SQL DBMSs.

** Gellish implementation is possible with any DBMS (Gellish, n.d.)

Legend: ● Ready
○ Not Ready / Uknown
◍ Ready, but not as existing plug in

[22] XML, Extensible Markup Language, describes a class of data objects including a partial description of the behavior of computer programs which process XML (Bray, Paoli, Sperberg-McQueen, Maler, & Yergeau, 1998).
[23] JSON, JavaScript Object Notation, is an easy to read and write data exchange format, based on Java (Crockford, 2013).
[24] YAML, "YAML Ain't Markup Language," is a human – friendly data serialization language, portable between programming languages (Ben-Kiki, Evans, & Ingerson, 2009).
[25] Gellish, Generic Engineering Language, is a modeling language for data storage (Henrichs, 2009).
[26] CSV, Comma Separated Values, is a file format used for data exchange. CSV is supported by spread-sheet application, such as Microsoft Excel (Edoceo, n.d.).
[27] Black marked are natively supported, grey are supported with external implementation. This table is based on several sources (Adams, 2014; Boncz, Manegold, & Rittinger, 2005; Gellish, n.d.; Intellidimension, n.d.; Microsoft, n.d.-a, n.d.-b, n.d.-e; Minh Duc, 2013; MonetDB, n.d.-d; MySQL, 2015; PostgreSQL, n.d.; W3C, 2003).

3

Preparing the Data Set for a CAD – GIS Environment

This chapter starts by analyzing the compatibility possibility of CAD and GIS in section 3.1 with help of the gained knowledge of chapter 2. With help of this analysis the data set of Fugro GeoServices will be prepared for a CAD – GIS environment in section 3.2. Section 3.2 also includes an analysis of the data set of Fugro GeoServices.

3.1 CAD – GIS Compatibility Analysis

For enabling the storage of CAD geometries in a 3D spatial DBMS it is evident to analyze the similarities between CAD and GIS. From chapter 2 a CAD – GIS potential table can be made, as seen in table 3.1. Table 3.1 shows that Oracle and PostgreSQL are the only DBMSs which support 3D geometry. MonetDB is currently not supporting 3D data, but will support 3D point clouds and voxels in the future (MonetDB, n.d.-b). Boundary representations (including polyhedra), point clouds and parametric design are natively supported by DBMSs. TEN and NURBS storage is possible in Oracle, because Wesselingh (2007) and Pu (2005) have implemented TENs and NURBS in Oracle during their MSc. Thesis. However this is not part of the Off the shelf product and is marked grey in table 3.1. Both Oracle and PostgreSQL support TENs when the boundary surface is triangulated and the geometry is saved as a boundary solid. Parametric design is possible with Oracle and PostgreSQL, because both DBMSs allow users-defined types.

Table 3.1: CAD-GIS potential table. This CAD – GIS potential analysis has been made by combining information from chapter 2. The 2D and 3D ability of the DBMSs are compared with the CAD and GIS storing methods, formats and solid types.

	Dimension ready		Method ready								
DBMS	2D	3D	Polyhedra	NURBS	Voxels	Octrees	Point Cloud	Boundary Rep.	TEN	3D Voronoi Diagram	Parametric Design
Oracle	●	●	●	◉	○	○	●	●	▨	○	●
PostgreSQL	●	●	●	○	○	○	●	●	▨	○	●
MonetDB	●	◉	○	○	◉	○	◉	○	○	○	○
MS SQL	●	○	○	○	○	○	○	○	○	○	○
MySQL	●	○	○	○	○	○	○	○	○	○	○
Format											
DGN			●	○	○	○	●	●	○	○	●
DWG*			●	●	○	○	●	●	○	○	●
IFC			●	●	○	○	○	●	○	○	●
Solid type											
True			●	○	●	●	○	○	●	●	●
Boundary			●	●	○	○	●	●	●	○	●

* Bentley MicroStation enables only boundary solids in DWG.

Legend: ● Ready

○ Not Ready / Uknown

◉ Ready in future / Ready, but not as existing plug in

▨ Combination of the three above (color of background and color of stripes)

3.1.1 Formats versus methods

In table 3.1 the CAD and BIM formats of chapter 2 are compared with the methods for visualizing 3D spatial data. Table 3.1 shows that CAD formats support NURBS, point clouds, boundary representation and parametric design. The IFC BIM format is based on a boundary representation, NURBS and parametric design.

The data set of Fugro GeoServices is a native CAD data set. In order to store this data as 3D spatial data in a 3D spatial DBMS (Oracle or PostgreSQL) it is convenient to choose from one of these CAD methods (NURBS, point clouds, boundary representations and parametric design) to implement. However these methods are assigned to a boundary solid and/or a true solid.

3.1.2 True solids versus boundary solids

In table 3.1 a distinction has been made between true solids and boundary solids. Polyhedra, NURBS, point clouds and boundary representations are noted as a boundary solid, because these methods only describe the boundary of a 3D object. However point clouds only describe the boundary with points, these points can be interpolated for describing the whole boundary. However this is not a trivial thing to do in practice. Voxels, octrees, TEN and 3D Voronoi Diagram are noted as a true solid. Parametric design can be regarded as both.

Voxels, octrees, TEN and 3D Voronoi Diagrams are 3D repeated elements: One geometry is described by multiple small 3D geometries. Together these 3D repeated elements model a volume: a true solid. The small elements solely are boundary solids, but all these small elements combined are considered as a true solid, because the volume concept can be retrieved by the assumption that all boundary solids together form a true solid. Parametric design can be both true solid and boundary solid. This depends on the method of parametric design (parametric variation -, parametric combination -, parametric hybrid model), which is either a boundary solid or a true solid.

The focus of this thesis is on storing true solids in a 3D spatial DBMS. From the selected methods, supported by CAD, only parametric design is a true solid. However polyhedra resembles parametric design (figure 3.1 and 3.2). Polyhedra can be seen as a true solid when polyhedra are modelled with the constraint that holes in the faces of the polyhedra are only accepted when these holes are also filled with inner walls, as proposed by Kazar et al. (2008)(figure 3.1). By preserving the constrains of Kazar et al. (2008) this kind of solid modeling with polyhedra is similar to solid modeling with parametric combination modelling (figure 3.2): set theoretic solid modelling (section 2.4). The SFCGAL extension of PostgreSQL enables boolean operators on polyhedra. Performing boolean operations on polyhedral might result into an invalid solid, because the shell of the solid might become unconnected. When boolean operators are used for polyhedra the constrains as proposed by Ledoux (2013) has to be maintained for the validation of polyhedra: using Nef polyhedra to validate solids.

Another opportunity for converting polyhedra to a true solid is by representing the interior of the polyhedron as a set of TENs, as proposed by Penninga and Van Oosterom (2008).

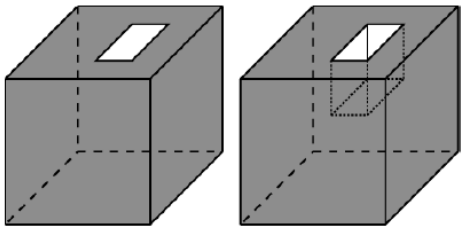


Figure 3.1: Invalid solid polyhedron (left, boundary solid) versus valid solid polyhedron (right, true solid) (Kazar et al., 2008).

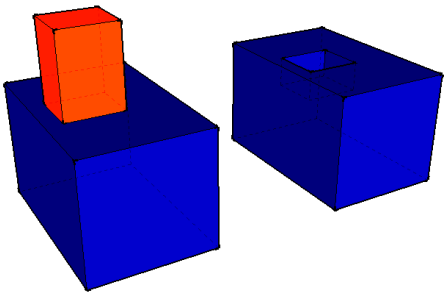


Figure 3.2: Parametric combination model, CSG. By differencing two geometries (red and blue) with a Boolean operator a similar geometry is retrieved as proposed by Kazar et al. (2008)(figure 3.1).

3.1.3 The link between CAD and GIS

The native CAD data set of Fugro GeoServices is based on parametric combination modelling, which resembles solid polyhedral modeling (when the constrains as proposed by Kazar et al. (2008) and Ledoux (2013) are maintained). With this link parametric design and solid polyhedral modeling are the desirable method for storing a true solid CAD geometry in a 3D spatial DBMS. With the SFCGAL extension of PostgreSQL it is possible to perform Boolean operations on polyhedra, which resembles parametric combination modelling.

For creating a 3D DBMS, based on parametric design and solid polyhedral modeling, it is necessary to create a parametric library to reconstruct these solid CAD geometries with polyhedra. Saving some topological attributes may improve the reconstruction, because then the relation between the 3D spatial objects is stored explicitly in a table (Brugman, 2010). This can improve the query execution time for e.g. buffering the data set around one geometry, because the flow of this geometry can be followed.

3.1.4 Conclusion

From the CAD – GIS potential analysis (table 3.1) can be concluded that the following methods are suitable for storing CAD geometries in a 3D spatial DBMS:

- Polyhedra
- Point clouds;
- Boundary representations;
- Parametric design.
-

From these CAD geometries parametric design and solid polyhedral modeling are the only methods for storing true solids in a 3D spatial DBMS. The main focus of this thesis is parametric design and solid polyhedral modelling, because these two methods resemble.

3.2 Data Set Preparation

When a 2D DBMS is upgraded to a 3D DBMS it is important to maintain the 2D data, because it is easier to provide an overview of the whole or part(s) of the data set in 2D than in 3D. In order to keep this characteristic the 3D data will be added as an extension to the current DBMS structure for enabling 3D spatial analyses. Keeping this notion in mind the following technical and function requirements has been set for the new 3D DBMS design of Fugro GeoServices:

- 3D query possibility;
- Result must be visualized in 2D and 3D;
- Volume calculation possibility;
- Buffer calculation (of a certain area);
- Spatial relationship calculation;
- Length calculation;
- Distinguish properties of the components, e.g. current state;
- Topology determination (no high priority).

The 3D DBMS is made with PostgreSQL, because the SFCGAL extension of PostgreSQL enables 3D modeling and performing Boolean operations with polyhedra.

3.2.1 From Point Cloud to Data Set

The data set of Fugro GeoServices is generated by processing the surveyed point cloud of the petrochemical industry. This point cloud is processed by manually fitting 3D solids in the point cloud. This manual processing is shown in figures 3.3 and 3.4. After combining separate point clouds to one point cloud with help of GCPs (Ground Control Points), the terrain can be modelled in 3D with help of solids. This modelling is done with piping software that saves the 3D model as a COE-file[28], which can be exported to a CAD-file. Processing the point cloud to a solid is done

[28] COE-files contain a Block RAM initialization code for specialized applications and development environments (File-Extensions.org, n.d.).

manually by separating the points, which define geometry. These separate points can be processed automatically to a solid by the software, or drawn by hand by inserting a solid by extruding a surface to a solid. Every 3D model is inserted back into the object space, until a 3D model is created with the point cloud as reference.

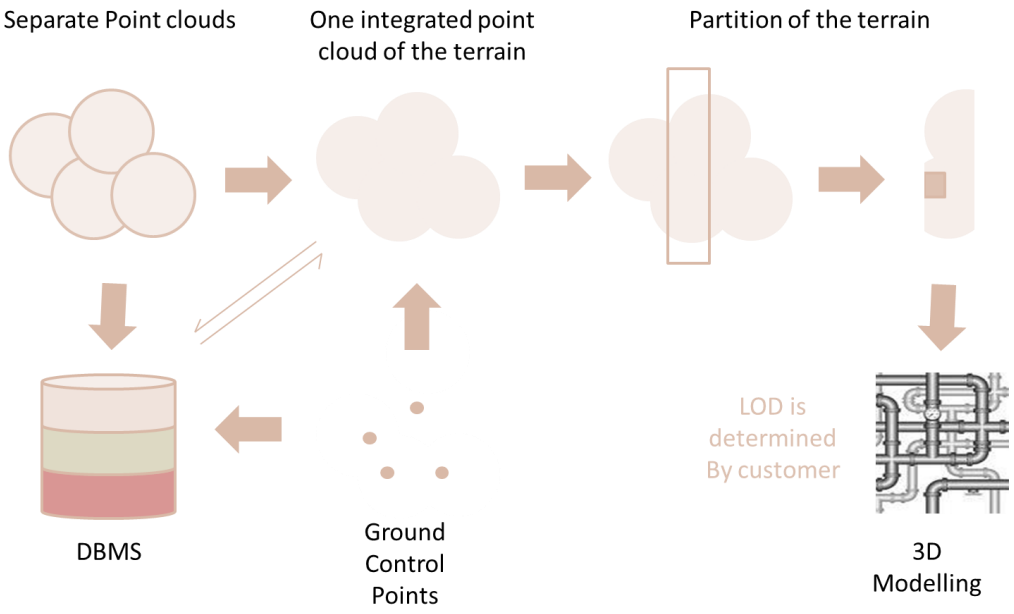


Figure 3.3: Point cloud processing, from processing separate point clouds to a 3D model.

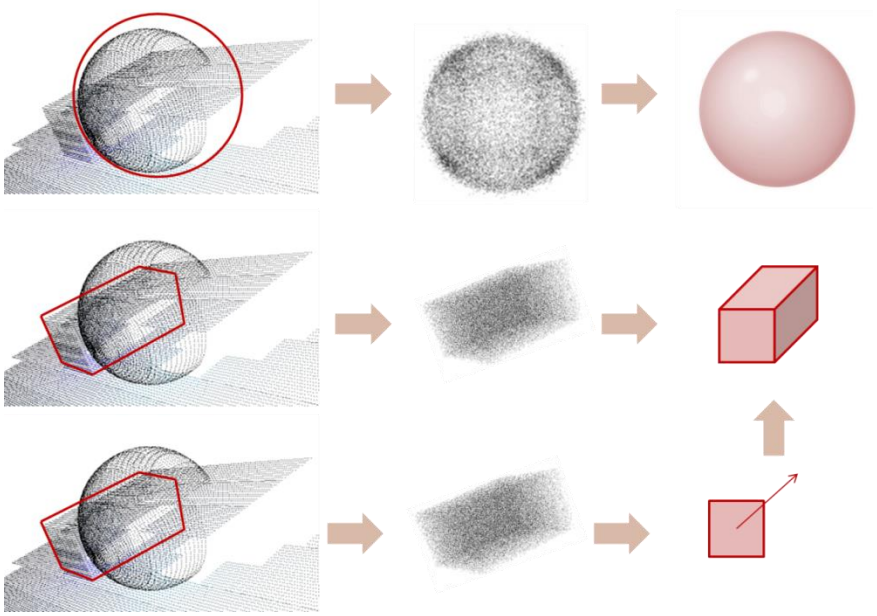


Figure 3.4: Point cloud processing, from point cloud to solids (from above to below: automatic solid generation of a sphere, automatic solid generation of a rectangle, and extruded solid generation of a rectangle).

3.2.2 Data Set Analysis

The data set of the petrochemical industry of Fugro GeoServices (figure 3.5) is georeferenced to a local CRS (Coordinate Reference System). This local CRS differs per data set and has been created in order to optimize the performance of the data set in the DBMS. This optimization has been made to have the largest amount of pipes in the model perpendicular to the X- and Y-axis. Therefore this local CRS is only a rotation of the Amersfoort RDNAP CRS (figure 3.6).

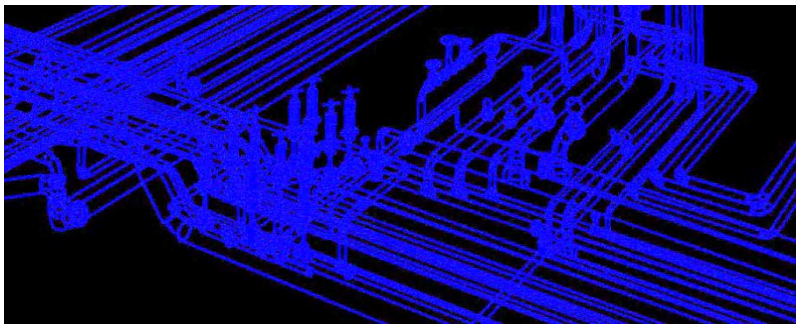


Figure 3.5: Pilot data set of Fugro GeoServices. This figure illustrates the 3D model, built in CAD, of the petrochemical industry.

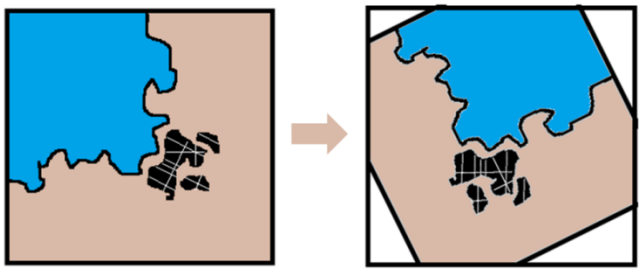


Figure 3.6: Simple illustration of the rotated local CRS. Amersfoort RDNAP CRS (left) versus local CRS of the data set of the petrochemical industry (right).

For this graduation project the biggest data set of Fugro GeoServices will be used: the petrochemical industry in Pernis. The data set contains parametric combination models and is saved as a CAD-file (DGN format) in Bentley MicroStation. Each geometry in the CAD-file (e.g. pipes, mechanicals, basemap and structures) is a solid geometry (primitive solid) in a cell. Table 3.2 shows the division of the different geometry types of the data set of Pernis. The parametric attributes of the geometries (e.g. radius, center point) are stored as characteristics of 2D lines in a spatial DBMS (figure 3.7). These parametric attributes are formatted as a Pipe JSON. The current DBMS of Fugro GeoServices does not store the topology of the geometries and has only indexed the id of the geometries.

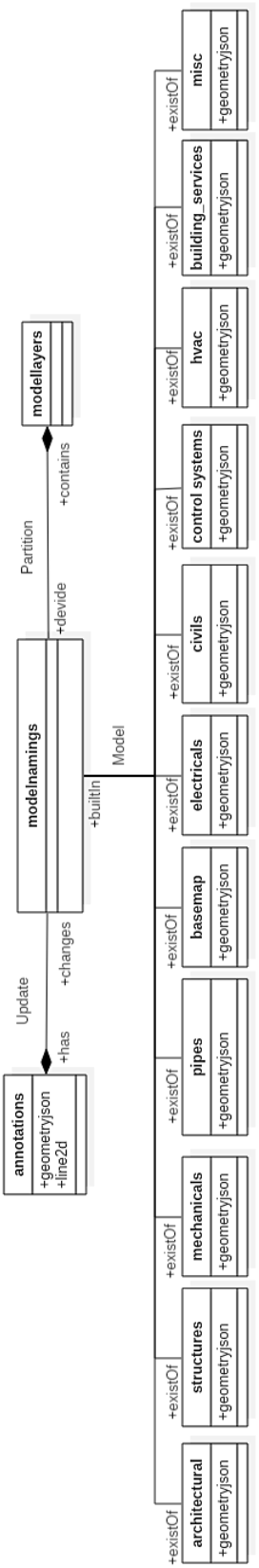


Figure 3.7: UML diagram[30] of the current DBMS structure of Fugro GeoServices. This UML diagram only emphasizes the geometrical attributes and the relations between classes.

[30] For the extensive UML diagram one is referred to Appendix A.

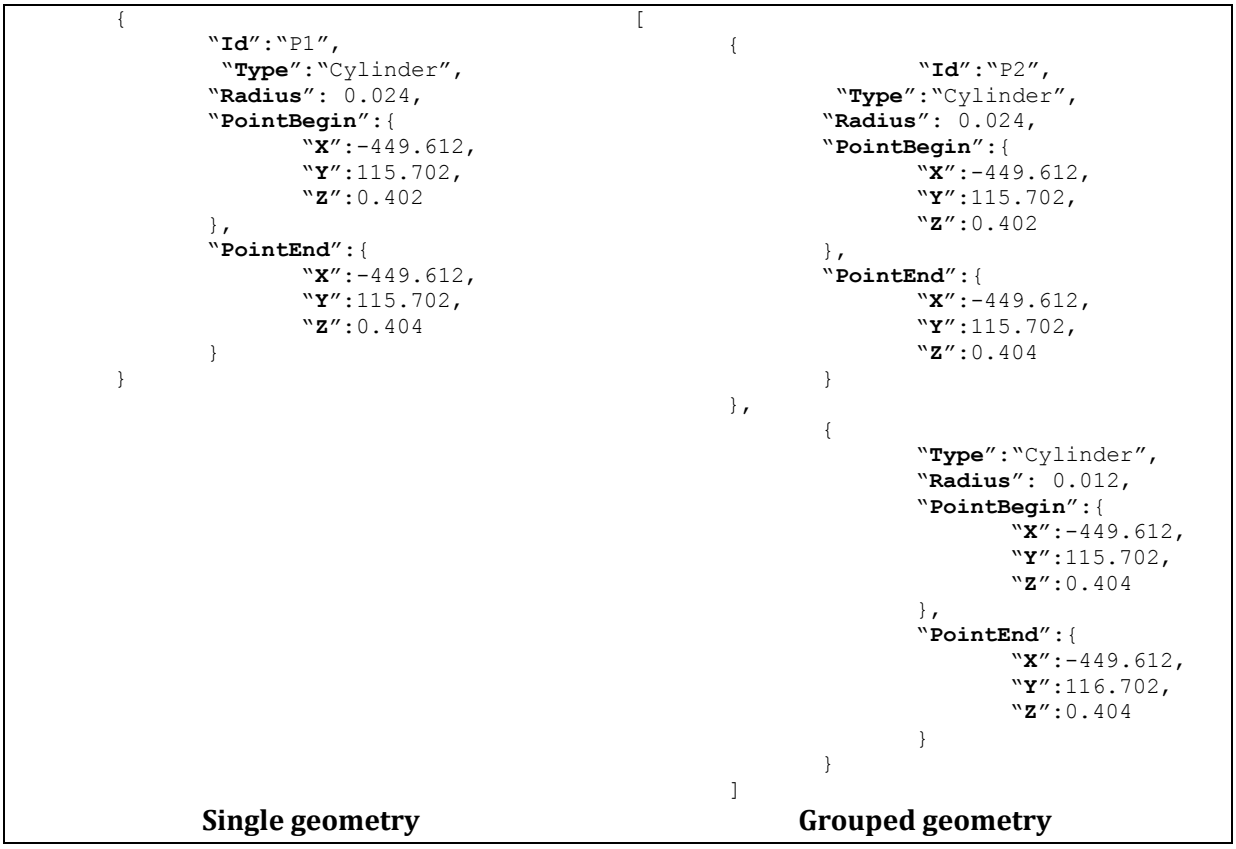
Table 3.2: Division of geometry types in the test data set of Pernis.

Geometry type	Amount in Data set
Cone	65058
Cylinder	195099
Dome	1275
Elbow	54051
SolidSphere	7248

3.2.3 Pipe JSON adaption

The generated CAD-files, containing parametric combination models, are processed with a program to a Pipe JSON. This Pipe JSON is a JSON (JavaScript Object Notation) structure developed by Fugro GeoServices for saving the attributes of the captured geometry from the DGN-file into a spatial DBMS (Database Management System). JSON is an alternative to XML for data exchange between server and external applications in human-readable text with help of attribute-value pairs. Pipe JSON is used as attributes of a 2D line in the spatial DBMS.

The first step to a 3D DBMS starts by adapting the current Pipe JSON format (figure 3.8) to an adapted Pipe JSON format[31] (figure 3.9) in order to ease the query ability of the DBMS.



```
{
        "Id":"P1",
         "Type":"Cylinder",
        "Radius": 0.024,
        "PointBegin":{
                "X":-449.612,
                "Y":115.702,
                "Z":0.402
        },
        "PointEnd":{
                "X":-449.612,
                "Y":115.702,
                "Z":0.404
        }
}
```

Single geometry

```
[
        {
                "Id":"P2",
         "Type":"Cylinder",
        "Radius": 0.024,
        "PointBegin":{
                "X":-449.612,
                "Y":115.702,
                "Z":0.402
        },
        "PointEnd":{
                "X":-449.612,
                "Y":115.702,
                "Z":0.404
        }
},
        {
                "Type":"Cylinder",
                "Radius": 0.012,
                "PointBegin":{
                        "X":-449.612,
                        "Y":115.702,
                        "Z":0.404
                },
                "PointEnd":{
                        "X":-449.612,
                        "Y":116.702,
                        "Z":0.404
                }
        }
]
```

Grouped geometry

Figure 3.8: Pipe JSON format. The extracted CAD parameters are formatted in Pipe JSON as an attribute of a 2D line in the current DBMS structure.

[31] One is referred to Appendix B for the adaption script of the Pipe JSON format.

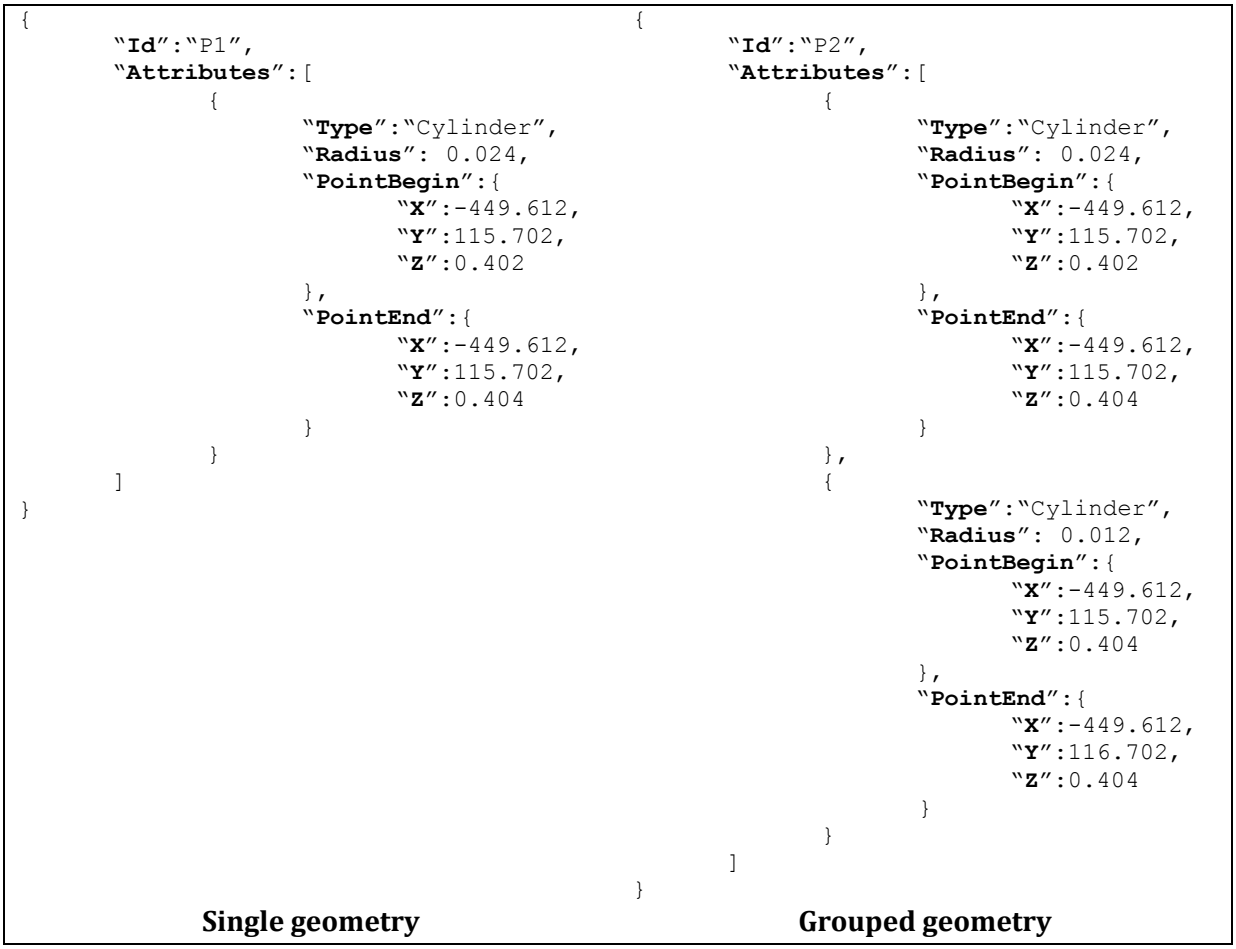


Figure 3.9: Adapted Pipe JSON format. The extracted CAD parameters are reformatted to the adapted Pipe JSON format in order to ease the query ability of the DBMS.

The adapted Pipe JSON format is similar to the current Pipe JSON, but differs in terms of formatting. By comparing the current Pipe JSON format with the adapted Pipe JSON format it is noticeable that the biggest change in both formats is in splitting the attributes of the geometries into 'Id' and 'Attributes.' This simple change eases to find specific attributes of the geometries, which is needed for doing analyses with the data set and reconstructing the geometries in 3D.

3.2.4 DBMS Design

Combining the knowledge of section 3.1 (CAD – GIS compatibility analysis) with the data set analysis a new DBMS design can be made. This new DBMS design is an extension of the current DBMS and is based on the adapted Pipe JSON format. In the current DBMS design the 2D line is the only geometrical component. The bounding box row of the pipes table is empty. This means that the 3D data is only stored in the Pipe JSON format (geometryjson). For the new DBMS design a 3D table is added to all geometry tables in the DBMS (figure 3.10), based on extracting the 3D data from the Pipe JSON format.

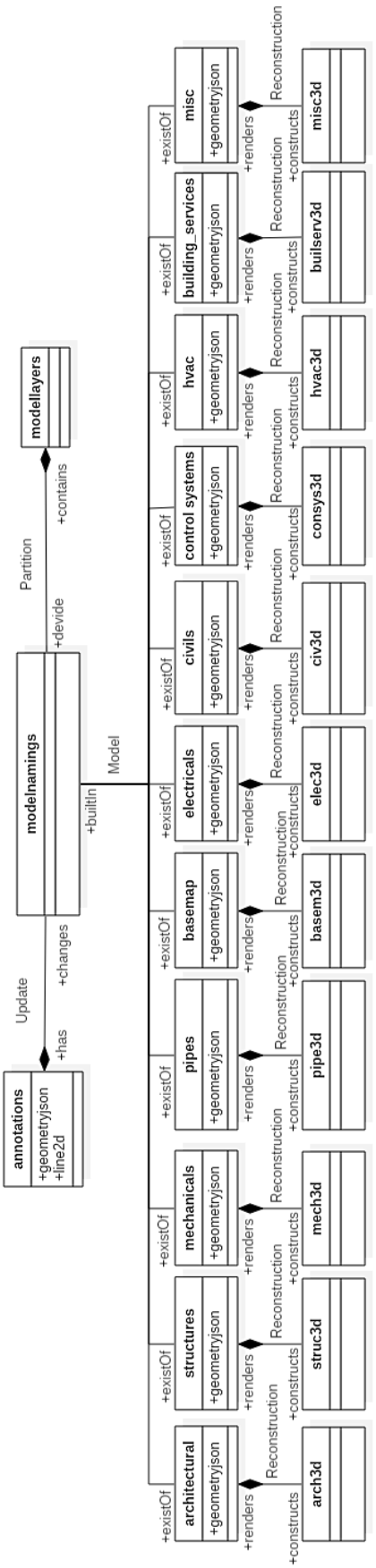


Figure 3.10: UML diagram of the new DBMS.

In this additional table all single and grouped geometries are extracted to single geometries, referencing to the primary key of the native geometry table. Extracting the grouped geometry to single geometries is simply done by flattening the nested JSON, as shown in figure 3.11. In order to maintain the bidirectional flow between DBMSs and GIS analyses a deliberate decision has been made to not throw away data from the Pipe JSON format, but only extending it and keeping the storage space as low as possible.

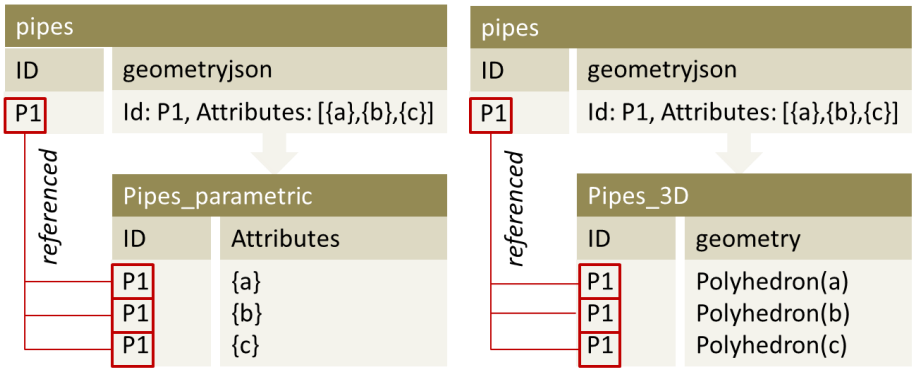


Figure 3.11: Extracting grouped geometry to single geometries by flattening the nested JSON. The single geometries are referenced to the primary key of the grouped geometry.

The focus of this thesis is to create a 3D DBMS, based on parametric design or solid polyhedral modeling. For comparing the performance of both methods two tables are added to the pipes geometry table of the DBMS (figure 3.12):

- 3D table (storing the 3D solid as a polyhedron);
- 3D parametric table (storing the 3D model parametrically with the adapted Pipe JSON).

This graduation project is focusing on the reconstruction of one geometry type: the cylinder, which is the largest group in the data set (table 3.2). The other geometry types are left out of this thesis, but are similar in reconstruction once they need to be implemented. This graduation project will compare the performance of having a 3D parametric DBMS (parametric design) versus a 3D non-parametric DBMS (solid polyhedral modelling). This performance comparison will only be done for all cylinder pipes in the pipes-table of the DBMS. The performance of all other geometries in all geometry tables of the DBMS is considered to be comparable to this comparison (3D parametric versus 3D non-parametric).

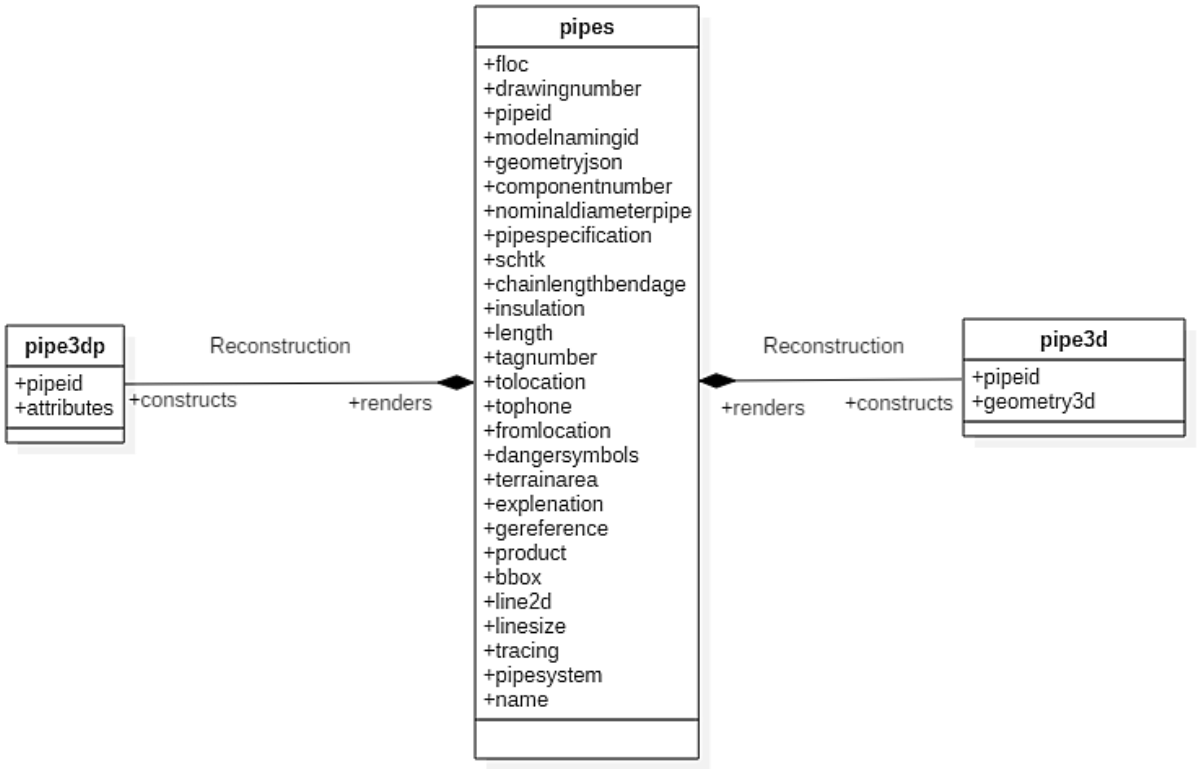


Figure 3.12: Additional tables (pipe3d (storing the 3D solid as a polyhedron) and pipe3dp (parametric design, based on solid polyhedral modeling)) in the current DBMS for comparing the performance of both methods.

4

Representing 3D Solids in a DBMS

This chapter focusses on the representation of 3D CAD solids in a 3D spatial DBMS. The 3D CAD solids are reconstructed in section 4.1. Section 4.2 will focus on the comparison of having a 3D parametric DBMS (parametric design) versus a 3D non-parametric DBMS (solid polyhedral modeling) in terms of a performance analysis. The performance of this 3D parametric DBMS will be optimized as much as possible in section 4.3. Section 4.4 concludes which 3D parametric DBMS performs best for comparing the scalability of a 3D parametric DBMS with the scalability of a 3D non-parametric DBMS in chapter 5.

4.1 3D Reconstruction of Solid Geometry

With the adapted Pipe JSON format it is possible to reconstruct the geometries in 3D. This 3D reconstruction enables to store the reconstructed geometries in 3D or to make a function for reconstructing the geometries with help of the Pipe JSON format in the DBMS. The following parameters are saved in the DBMS for cylinder geometries:

- the radius;
- the starting point, defined in X, Y and Z-coordinates;
- the ending point, defined in X, Y and Z-coordinates.

The reconstruction of the cylinder starts by buffering the starting point as a center point with the radius, which results into a circular polygon (figure 4.1). In order to retrieve the cylinder the circular polygon is extruded towards the ending point (figure 4.2). This circular polygon is approximated with 32 points, which has a deviation of 2,4 cm with the biggest cylinder pipe (radius equal to 0,5 meter) in reality. This deviation has been computed with the Sagitta (Eq. 4.1) and is caused by having no arcs, but straight lines between all points.

$$s = r\,(1 - \cos\left(\frac{1}{2}\alpha\right)) \qquad \text{Eq. 4.1}$$

where:

s = Sagitta

r = ray length

α = angle

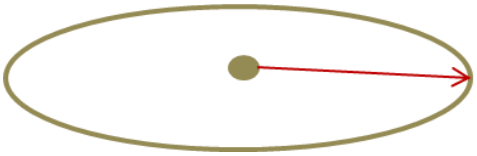


Figure 4.1: 3D Reconstruction step of the cylinder[32]. By buffering the starting point with the radius of the cylinder a circular polygon is reconstructed.

Figure 4.2: 3D Reconstruction step of the cylinder. By extruding the circular polygon (figure 4.1) towards the ending point the cylinder is reconstructed.

The normal of the reconstructed circular polygon is always perpendicular to the ground surface with no slope. When the ending point is not located straight above this circular polygon a rotation is necessary towards the ending point (figure 4.3). According to Kuipers (1999) normals have 2 rotations in 3D: a rotation along the X- or Y-axis (specifying the angle) and a rotation along the Z-axis (specifying the direction) (figure 4.3). Both rotations are angles which are computed with help of the dot product (Eq. 4.2).

$$A \cdot B = \left|\left|A\right|\right| \left|\left|B\right|\right| \cos\theta \qquad \text{Eq. 4.2.}$$

where:

A = Euclidian vector in direction a

B = Euclidian vector in direction b

||A|| = Magnitude of vector A

||B|| = Magnitude of vector B

θ = the angle between vector A and B

[32] One is referred to Appendix B for the function code (ST_CircelPoly).

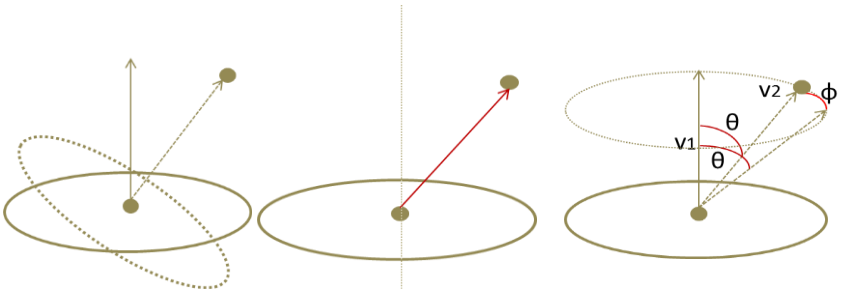


Figure 4.3: 3D Reconstruction step of the cylinder. By rotating the normal of the circular polygon towards the ending point the circular polygon is rotated with the same rotations of the normal.

These two rotations will result into a rotation angle and a rotation direction towards the ending point. Planes can be rotated with help of the rotation matrix (with help of the Euler angles) or quaternions (Kuipers, 1999). Both paths (rotation matrix and quaternions) have been investigated for rotating the circular polygon. The most convenient between both paths is rotating the circular polygon with help of the rotation matrix (Eq. 4.3), because this is a native PostgreSQL function.

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Eq. 4.3}$$

where:

R = Rotation in specified X-, Y-, or Z-direction

θ = Rotation or direction angle

Now the methodology for 3D reconstruction has been defined the 2D spatial DBMS[33] of Fugro GeoServices can be converted With PostgreSQL it is possible to convert the geometries to X3D-geometries with a 'built in' conversion. X3D, Extensible 3D, is the successor of VRML (Virtual Reality Modeling Language) and uses XML to express geometry (Brutzman & Daly, 2007). With help of these X3D-geometries it is possible to generate a X3D-document[34], which can be viewed by a X3D-viewer. This X3D-viewer must be able to handle large X3D-files. The generated X3D file had a size of 240 MB and could not be parsed by FreeWRL, because the X3D file was too big to parse. As a solution to this Octaga has been used, which was able to parse the large generated X3D-file. In figure 4.4 the reconstructed cylinder pipes of Fugro's data set is shown. Algorithm 4.1 shows the pseudocode of the reconstruction function, ST_3DReconstruct.

[33] The UML diagram of the current DBMS of Fugro GeoServices can be found in section 3.2.4 and Appendix A (extensive version).

[34] The SQL code for generating a X3D-document with PostgreSQL can be found in Appendix B.

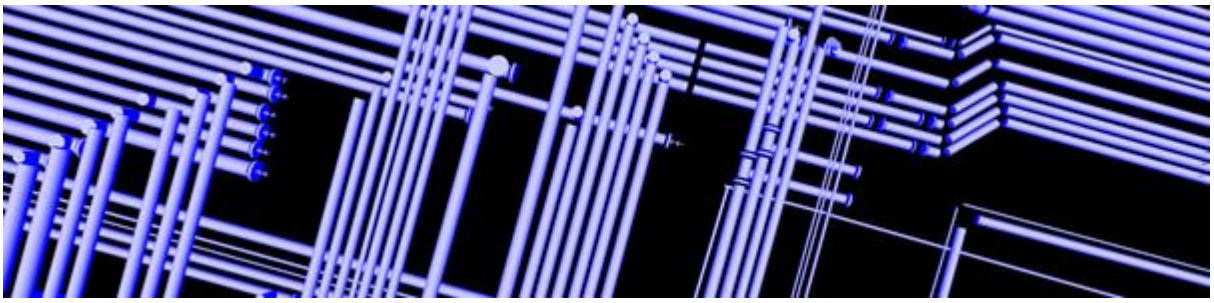


Figure 4.4: 3D reconstructed geometries (of type cylinder)[35] of Fugro's data set, visualized with Octaga X3D viewer. There are no connection between the pipes, because only the cylinder has been implemented (these connections are either elbows or cones).

```
Create function ST_3DReconstruct (point1, point2, radius) as
       polygon = point1.buffer(radius)

       If point1.x <> point2.x and point1.y <> point2.y <> 0 then:
              rotangle = Dotprod (point1, point2)
              polygon = polygon.rotate(rotangle)
              point1 = point1.rotate(rotangle)
              dirangle = Dotprod (point1, point2)
              polygon = polygon.rotate(dirangle)
       End if

       polyhedron = polygon.extrude(point2)
       return polyhedron;
```

Algorithm 4.1: Pseudocode for 3D Cylinder reconstruction function[36].

4.2 Comparing Parametric with Non-Parametric

For comparing a 3D parametric DBMS with a 3D non-parametric DBMS two tables are added to the pipes geometry in section 4.2.1. The performance of both tables is compared in a mini-benchmark setting, comparing the buffering performance in section 4.2.2, the intersection performance in section 4.2.3 and the area computation performance in section 4.2.4.

4.2.1 Creating the Parametric and Non-Parametric table

For creating the 3D parametric and 3D non-parametric tables all geometries are separated into single geometries and stored into a temporary table with algorithm 4.2. This temporary table is stored in a new created 3D parametric table: pipe3dp. This 3D parametric table refers to the original pipes table of the existing DBMS as a constraint. In the 3D parametric table only the geometries of type cylinder are stored. This process has been done with help of algorithm 4.3.

[35] One is referred to Appendix B for the function code, ST_3DCylinder.
[36] One is referred to Appendix B for the function code, ST_3DReconstuct (including the dot product function, __ST_DotProd). The extrude and buffer function are SFCGAL (ST_Extrude) and PostGIS (ST_Buffer) native functions.

```
CREATE TABLE temptable AS
SELECT pipeid, jsonb_array_elements(geometryjson->'Attributes') attributes
FROM geo3d.pipes;
```

Algorithm 4.2: Splitting all geometries into single geometries by extracting the attributes of the Pipe JSON format.

```
DROP TABLE IF EXISTS geo3d.pipe3dp;

CREATE TABLE geo3d.pipe3dp

(
   pipeid integer,
   attributes jsonb,
      CONSTRAINT "FK_pipe3dp_pipes" FOREIGN KEY (pipeid)
         REFERENCES geo3d.pipes (pipeid) MATCH SIMPLE
         ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
   OIDS=FALSE
);

INSERT INTO geo3d.pipe3dp
SELECT *
FROM temptable
WHERE (attributes->'Type')::text='"Cylinder"';
```

Algorithm 4.3: Generating the 3D parametric table of the pipes geometry table with only cylinder geometries. This 3D parametric table is referenced to the original pipes table.

However algorithm 4.2 seems unnecessary, it is important to first split all JSON objects into single objects before making a selection (selecting all cylinders only). When this selection is not needed (when all geometry types are implemented) algorithm 4.2 is not necessary. In this case algorithm 4.2 and 4.3 can be combined. With the generated 3D parametric table of the DBMS the 3D non-parametric table can be generated with help of algorithm 4.4, which is also referring to the original pipes table as a constraint. This constraint ensures that every pipe has a 3D geometry.

```
CREATE TABLE geo3d.pipe3d
(
   pipeid integer,
   geometry3d geometry,
   CONSTRAINT "3dim" CHECK (ST_ZMflag(geometry3d)=2),
   CONSTRAINT "Polyhedra" CHECK (GeometryType(geometry3d)= 'POLYHEDRALSURFACE'),
   CONSTRAINT "FK_pipe3dgeo_pipes" FOREIGN KEY (pipeid)
      REFERENCES geo3d.pipes (pipeid) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

```
WITH (
   OIDS=FALSE
);

INSERT INTO geo3d.pipe3d
SELECT pipeid, ST_3Dreconstruct(attributes)
FROM geo3d.pipe3dp;
```

Algorithm 4.4: Generating the 3D non-parametric table of the pipes geometry table with reconstructed cylinder geometries only. This 3D non-parametric table is referenced to the original pipes table.

In order to optimize the performance of both 3D parametric - and 3D non-parametric table a spatial index has to be created on both tables. For the 3D non-parametric table a GIST-index is created, which uses a 3D bounding box as pre-filter (Leslie, 2009). This GIST-index is created with algorithm 4.5.

```
CREATE INDEX pipe3dgeo_index
  ON geo3d.pipe3d
  USING gist
  (geometry3d);
```

Algorithm 4.5: Creating a GIST-index in PostgreSQL on the 3D geometries of the 3D non-parametric table.

For creating a spatial index on the 3D parametric table a function based index has to be made. The 3D parametric table is based on a JSON, which is non-spatial. Creating a GIST-index is not applicable on a non-spatial table, unless a function can generate spatial data with help of the stored JSON-attributes. By making a function which generates a spatial bounding box, based on the 3D center line and radius of the cylinder, a spatial function based index can be created (algorithm 4.6). By differencing the radius from the minima X-, Y- and Z-values and adding the radius to the maxima X-, Y- and Z-values of the 3D centerline a 3D bounding box can be generated (figure 4.5, algorithm 4.7). The generated bounding boxes are reconstructed too big on purpose in order to have a proper bounding box with a low calculation cost. By generating a bounding box according to algorithm 4.7 all cylinders, rotated and not rotated, fit the bounding box.

```
CREATE INDEX pipe3dp_index
  ON geo3d.pipe3dp
  USING gist
  (ST_bboxline(attributes));
```

Algorithm 4.6: Creating a function based GIST-index in PostgreSQL on the 3D parametric table.

```
Create function ST_bboxline(Radius, PointBegin, PointEnd) as
       centerline = Line(PointBegin, PointEnd)

       xma = centerline.xmax + Radius
       yma = centerline.ymax + Radius
       zma = centerline.zmax + Radius

       xmi = centerline.xmin - Radius
       ymi = centerline.ymin - Radius
       zmi = centerline.zmin - Radius

       bbox = 3DBox( Point(xmi, ymi, zmi), Point (xma, yma, zma))
       return bbox;
```

Algorithm 4.7: Pseudocode for generating a 3D bounding box (ST_bboxline[37]).

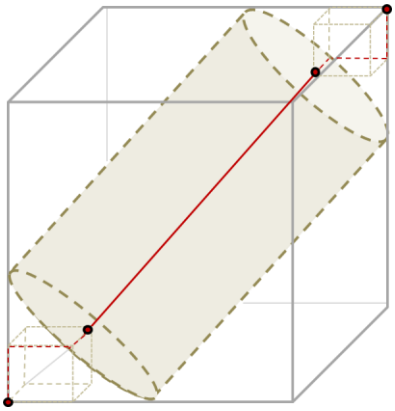


Figure 4.5: Generating a 3D bounding box (ST_bboxline), based on the minima and maxima X-, Y-, and Z-values of the 3D center line and radius of the cylinder.

By having an indexed 3D parametric table and an indexed 3D non-parametric table the performance of both tables can be compared in a mini-benchmark setting[38]. This mini-benchmarking is used as a test phase for the benchmark design, as proposed by Van Oosterom et al. (2015).

For the mini-benchmarking three query conditions are compared (figure 4.6):

1. Pipes within a certain region at a specified point (figure 4.6a).
   Calculating the amount of pipes within a region of 5 meter at a random chosen point (algorithm 4.8).

[37] One is referred to Appendix B for the function code, ST_bboxline.

[38] For the queries of the mini-benchmark one is referred to Appendix B.

2. Intersection of pipes at a specific region (figure 4.6b).

   Calculating the amount of intersecting pipes within the calculated buffer of query condition 1. Initially this query conditions was meant to be performed on the whole data set. However this query condition has been adapted to intersections within the computed buffers only, because the query execution time for the whole data set took too long.

3. The surface area of a cylinder (figure 4.6c).

   Calculating the surface area of a cylinder. Initially it was intended to compare the volume computation of a 3D parametric DBMS with a 3D non-parametric DBMS. This comparison was not possible, because the SFCGAL extension of PostgreSQL has not implemented the volume calculation of 3D objects.

```
Create function randompointinpoly(geom, count) as
       Loop = 1
       Create List lst

       While loop != count:
              ranx = ((geom.xmax – geom.xmin) * random()) + geom.xmin
              rany = ((geom.ymax – geom.ymin) * random()) + geom.ymin
              ranz = ((geom.zmax – geom.zmin) * random()) + geom.zmin

              randompt = Point(geom.zmin) + ranx,geom.ymin + rany, geom.zmin + ranz)
              loop = loop + 1
              append lst

       return lst;
```

Algorithm 4.8: Pseudocode for generating random point.[39]

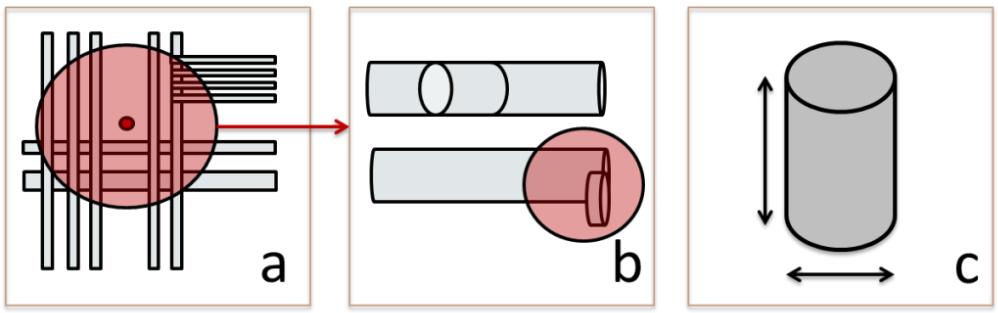


Figure 4.6: Illustration of query conditions, (from left to right) pipes within a certain region, intersection of pipes, surface area calculation.

[39] One is referred to Appendix B for the function code, randompointsinpolygon3d.

The performance will be measured in terms of hits (or area) and execution time. The execution time will be measured by performing the query with the 'Explain Analyze' option in PostgreSQL. For the mini-benchmark (data set of 231 MB, 19509900 objects) an Acer V3-571G laptop is used on the TU Delft test server with the following specifications:

- Operating System: Windows 8
- Processor: Inter Core i7
- Processor Cores: 4
- Maximum frequency: 3.2GHz
- PostgreSQL version: 9.4.1
- PostGIS version: 2.2.0dev
- SFCGAL version: 1.0.5

4.2.2 Buffering

The first query of the performance analysis is generating 100 random points somewhere within the data set (figure 4.7) with algorithm 4.9. These random points are buffered. All pipes which are entirely within these buffers are selected. The result of the 3D parametric table (algorithm 4.10) versus the 3D non-parametric table (algorithm 4.11) in terms of query execution time and hits is shown in table 4.1.

Table 4.1 shows that the query execution time of the 3D parametric DBMS is tremendously higher compared with the 3D non-parametric DBMS. This is caused by two things:

1. **Reconstructing the geometry parametrically is too heavy**. DBMSs are not efficient in performing heavy computations (Pu, 2005).
2. **The function based index is not used.** The query optimizer of PostgreSQL choses either a sequential scan or an indexed scan for executing a query. This choice is based on the type of query. When using a function based index in combination with a mathematical expression of function the sequential scan is used instead of the indexed scan (figure 4.7). PostgreSQL is first performing the function or mathematical expression sequentially, followed by comparing the outcome sequentially with the constraint. If PostgreSQL only has to compare the value of the function based index with another value the indexed scan is used instead of the sequential scan.

   By interpreting this knowledge to algorithm 4.11 a function based index can only be created on 'ST_bboxline(attributes)' in red. It is not possible to index 'ST_3DDWithin(ST_bboxline(attributes, rpoint, 5)' (in blue and red), because this query

statement depends on multiple table. Indexes cannot be created on values which are influenced from outside (e.g. index sum of two table values). For this reason it is not effective to create a function based index as a replacement for a spatial index.

The 3D parametric table needs a lot of improvement in terms of query execution time.

Table 4.1: Performance of the buffering benchmark, 3D parametric (3DP) versus 3D non-parametric (3D) in terms of query execution time and number of hits.

	Time (ms)	Amount of combinations*	Number of Hits	Number of pipes**	Time (ms) per pair***
3D	852	19509900	229	179	4,37e-05
3DP	9637491	19509900	229	179	0,494

* Amount of geometries multiplied by the number of random points.
** Number of pipes: number of hits, distinguished on pipe id.
*** The time per pair is the computation time per true and false hits.

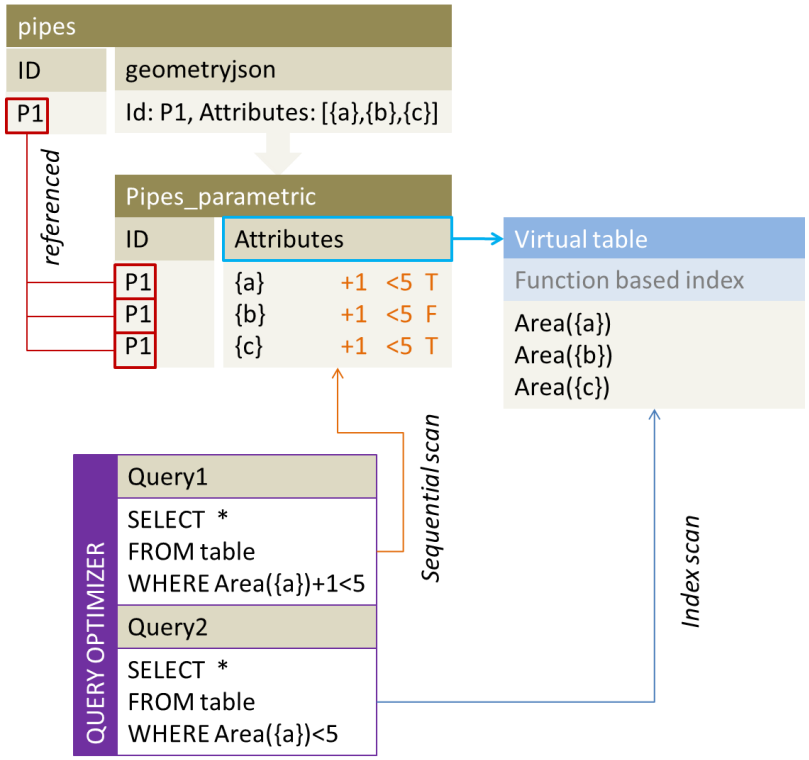


Figure 4.7: The behavior of PostgreSQL's query optimizer when having a function based index (using a sequential scan or a indexed scan).

Figure 4.8: Generated random points in the pilot data set of Fugro GeoServices (Pernis).

```
CREATE TABLE benchpoints as
SELECT randompointsinpolygon3d(st_3dextent(pipe3d.geometry3d), 100) AS rpoint
FROM geo3d.pipe3d;
```

Algorithm 4.9: Query for generating 100 random points somewhere within the data set with help of function RandomPointsInPolygon3d (algorithm 4.8). PostgreSQL's ST_3DExtent generates a bounding box the data set, having all geometries as input.

```
SELECT a.*
FROM geo3d.pipe3d a JOIN benchpoints ON ST_3DDWithin(geometry3d, rpoint, 5);
```

Algorithm 4.10: Buffering benchmark query for 3D non-parametric table.

```
SELECT a.*
FROM geo3d.pipe3dp a JOIN benchpoints ON ST_3DDWithin(ST_bboxline(attributes),
rpoint, 5)
WHERE ST_3DDWithin(ST_3Dreconstruct(attributes), rpoint, 5) is true;
```

Algorithm 4.11: Buffering benchmark query for 3D parametric table[40]. The red and blue phrase emphasizes the possible options for creating a function based index.

[40] In comparison with algorithm 4.10 a where clause is added in order to incorporate a second filter. This second filter checks if the geometries are also within the buffer (besides the bounding box).

4.2.3 Intersections

The second query of the performance analysis is computing all intersections of cylinders present within the computed buffers of section 4.2.2. Two kinds of intersection computations will be compared: the native PostgreSQL ST_3DIntersects function (which uses 3D geometry as input) and the parametric intersection computation (algorithm 4.12, figure 4.9). The parametric intersection computation is based on the assumption that if the shortest distance between the centerlines of two candidate cylinders is smaller than the sum of both radii, both cylinders are intersecting. This assumption is not always true if the shortest line between both centerlines is intersecting with the starting and/or ending point of the centerlines. In this case the cylinders are reconstructed in 3D for computing if both geometries are intersecting. If there is no intersection between the starting and/or ending point and the shortest line both cylinders are intersecting.

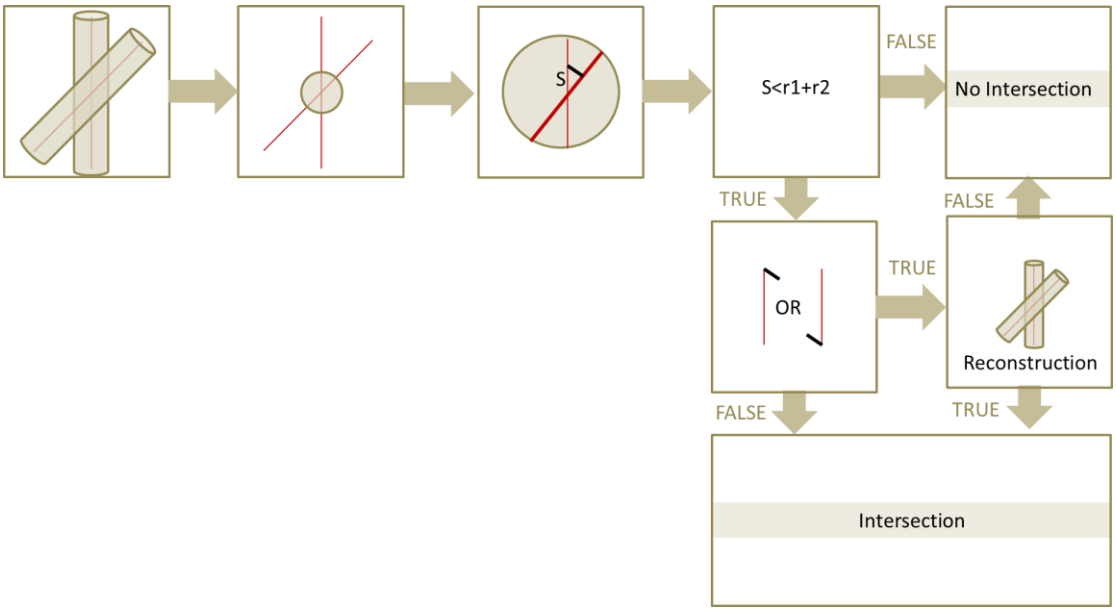


Figure 4.9: Flow of parametric intersection computation (algorithm 4.12). By comparing the shortest distance of the center lines with the sum of the radii there is no intersection or a possible intersection.

```
Create function ST_ParaIntersect (geom1, geom2) as
       Dis = ShortestDistance(geom1, geom2)

       If Dis < geom1.radius + geom2.radius then:
              sline = ShortestLine(geom1, geom2)

              If Intersection(sline, geom1.pointbegin) is true then:
                     3d1 = 3DReconstruct(geom1)
                     3d2 = 3DReconstruct(geom2)
                     return ST_3DIntersection(3d1, 3d2);
```

```
            End if;

            If Intersection(sline, geom1.pointend) is true then:
                  3d1 = 3DReconstruct(geom1)
                  3d2 = 3DReconstruct(geom2)
                  return ST_3DIntersection(3d1, 3d2);
            End if;

            If Intersection(sline, geom2.pointbegin) is true then:
                  3d1 = 3DReconstruct(geom1)
                  3d2 = 3DReconstruct(geom2)
                  return ST_3DIntersection(3d1, 3d2);
            End if;

            If Intersection(sline, geom2.pointend) is true then:
                  3d1 = 3DReconstruct(geom1)
                  3d2 = 3DReconstruct(geom2)
                  return ST_3DIntersection(3d1, 3d2);
            Else:
                  return true;
            End if;
      Else:
            return false;
      End if;
```

Algorithm 4.12: Pseudocode for parametric intersection computation[41].

In order to measure the query time of the intersection computation all pipes present within the buffer (based on the filter) of section 4.2.2 are saved in a temporary table. This temporary table is used for performing the intersection computations. The result of the 3D parametric table (algorithm 4.13) versus the 3D non-parametric table (algorithm 4.14) in terms of execution time and hits is shown in table 4.2.

Table 4.2 shows that the intersection computation in terms of query execution time is computed faster parametrically than with the native PostgreSQL intersection function. However there is a difference in the number of intersections (240 versus 0). This difference is caused by the reconstruction errors of reconstructing the geometries. This means that the intersection computation is a quality check of the reconstructed geometry. From the intersection computation can be deducted that the reconstruction algorithm has an error of 0,49%. Comparing table 4.2 with table 4.1 it is noticeable that the parametric computation is faster in computing intersections. This is caused by having cheap pre-filters, which limits the 3D reconstruction of geometries for intersection computations with geometries as input.

[41] One is referred to Appendix B for the function code, ST_ParaIntersect.

Table 4.2: Performance of the intersection benchmark, 3D parametric (3DP) versus 3D non-parametric (3D) in terms of query execution time and number of intersections.

	Time (ms)	Amount of combinations	Number of intersections	Intersecting pipes*	Time (ms) per pair**
3D	11736	51956	240	91	0,226
3DP	3332	51956	0	0	0,064

* Intersecting pipes: number of intersections, distinguished on pipe id.
** The time per pair is the computation time per true and false hits.

```
CREATE TABLE inter AS
SELECT a.*
FROM pernis.pipe3d a JOIN benchpoints ON ST_3DDWithin(geometry3d, rpoint, 5);

SELECT b.*
FROM inter a JOIN inter b ON a.pipeid <> b.pipeidWHERE
ST_3DIntersects(a.geometry3d,b.geometry3d) IS TRUE;
```

Algorithm 4.13: Intersection benchmark query for 3D non-parametric table.

```
CREATE TABLE interp AS
SELECT a.*
FROM pernis.pipe3dp a JOIN benchpoints ON ST_3DDWithin(bbox3d, rpoint, 5)
WHERE ST_3DDWithin(ST_3Dreconstruct(attributes), rpoint, 5) IS TRUE;

SELECT b.*
FROM interp a JOIN interp b ON a.pipeid <> b.pipeidWHERE
ST_ParaIntersect3(a.attributes,b.attributes) IS TRUE;
```

Algorithm 4.14: Intersection benchmark query for 3D parametric table.

4.2.4 Area Computation

Besides comparing the buffering performance and the intersection performance it is also interesting to compare the attribute calculation of the geometries. The only relevant implemented attribute calculation of a 3D geometry by PostgreSQL is the area computation, ST_3DArea. The parametric area computation is based on algorithm 4.15. Table 4.3 compares algorithm 4.15 with the 3D area computation of PostgreSQL, using a sample of pipes. Table 4.3 shows that the 3D area computation of PostgreSQL always gives the value 0. Apparently the 3D area computation function of PostgreSQL is not working well with extruded polyhedral surfaces. For this reason the parametric 3D area computation will not be compared any further with the 3D non-parametric area computation. The parametric area computation performs well in terms of query execution time compared to the 3D non-parametric area computation.

Table 4.3: Sample performance of the area computation, 3D parametric (3DP) versus 3D non-parametric (3D) in terms of query execution time and computed area.

Sample		Time (ms)	Area
1	3D	834	0
	3DP	27	0,002
2	3D	1419	0
	3DP	26	0,190
3	3D	1368	0
	3DP	26	0,065
4	3D	834	0
	3DP	27	0,020
5	3D	1110	0
	3DP	26	3,330

```
Create function ST_CylinderArea (Radius, PointBegin, PointEnd) as
   Area_Circel = Radius^2 * pi()
   Area_SideSurface = Radius * 2 * pi() * Length (PointBegin, PointEnd)
   CylinderArea = Area_Circel + Area_SideSurface
   Return CylinderArea;
```

Algorithm 4.15: Pseudocode for parametric area computation of a cylinder[42].

4.3 Parametric Optimization Solutions

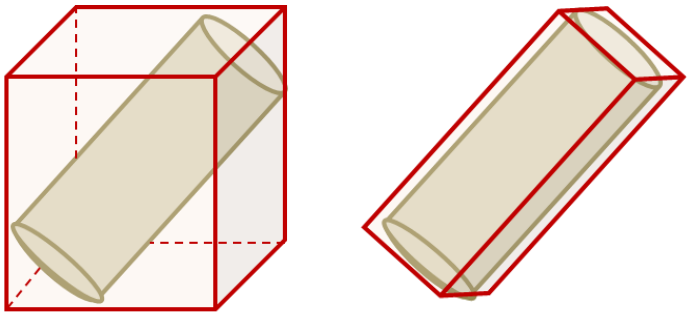
From section 4.2 can be concluded that the buffering performance of a 3D parametric DBMS (with a function based index) needs to be improved: In table 4.1 can be seen that the 3D parametric DBMS (3DP) is much slower than the 3D non-parametric DBMS (3D). Section 4.1 has shown that creating a function based index as replacement for a spatial index is not efficient for all computation purposes. This section will focus on optimizing the performance of a 3D parametric DBMS by storing a 3D approximation of the geometry for creating a spatial index. The following 3D approximations are considered (figure 4.10):

- 3D bounding box (algorithm 4.16);
    - Stored as a polyhedron (spatial indexed, GIST)(3DP2);
    - Stored as PostgreSQL's Box3D (allowing no index)(3DP3).

      An alternative for saving a 3D bounding box as a polyhedron is PostgreSQL's Box3D geometry. A trade-off of PostgreSQL's Box3D is that there is no possibility for creating an index on this geometry type.

[42] One is referred to Appendix B for the function code, ST_CylinderArea.

- 3D rotated bounding box (algorithm 4.16).
    - Stored as a polyhedron (spatial indexed, GIST)(3DRBB).



3DP2 & 3DP3 **3DRBB**

Figure 4.10: 3D approximations of a 3D object, 3DP2[44] (3D bounding box as a polyhedron) & 3DP3[45] (3D bounding box as Box3D) versus 3DRBB[46] (3D rotated bounding box).

```
Create function ST_rbb (Radius, PointBegin, PointEnd) as
       bpx = PointBegin.x
       bpy = PointBegin.y
       bpy = PointBegin.z

       epx = PointEnd.x - PointBegin.x
       epy = PointEnd.y - PointBegin.y
       epy = PointEnd.z - PointBegin.z

       pt1 = bpx - Radius, bpy + Radius, bpz
       pt2 = bpx + Radius, bpy + Radius, bpz
       pt3 = bpx + Radius, bpy - Radius, bpz
       pt4 = bpx - Radius, bpy - Radius, bpz

       poly = CreatePolygon(pt1, pt2, pt3, pt4, pt1)
       poly = Rotate(poly, PointEnd)
       poly = Extrude(poly, epx, epy, epz)

       return poly;
```

Algorithm 4.16: Pseudocode for creating a 3D rotated bounding box (3DRBB).

The buffering performance of storing these 3D approximations is shown in table 4.4. Table 4.4 shows that saving a 3D approximation of a 3D geometry improves the query execution time. The number of hits of PostgreSQL's Box3D geometry (3DP3) is negative effected, because the Box3D data type is converted to a 2D polygon for performing queries.

[44] One is referred to Appendix B for the function codes, ST_3DCylinderbb (depends on ST_CircelPolybb).
[45] One is referred to Appendix B for the function codes, ST_bbox.
[46] One is referred to Appendix B for the function codes, ST_Rbb

Table 4.4: Performance of the buffering benchmark[48], based on storing a 3D approximation of the 3D geometry as a 3D bounding box (polyhedron)(3DP2), 3D bounding box (box3d)(3DP3) and 3D rotated bounding box (polyhedron)(3DRBB). Compared is the query execution time and number of hits.

	Time (ms)	Amount of combinations*	Number of Hits	Number of pipes**	Time (ms) per pair***
3DP2	889	19509900	229	179	4,56e-05
3DP3	21030	19509900	227	177	0,001
3DRBB	737	19509900	229	179	3,78e-05

* Amount of geometries multiplied by the number of random points.
** Number of pipes: number of hits, distinguished on pipe id.
*** The time per pair is the computation time per true and false hits.

4.4 Comparing all Parametric Solutions

As a result of the bad query execution time of the 3D parametric DBMS in section 4.2 (for buffering pipes within a certain region), several methods have been explored for optimizing the query execution time in section 4.3 (figure 4.11). Before arguing which method optimizes the 3D parametric performance the best, the storage size of all methods is presented in table 4.5.

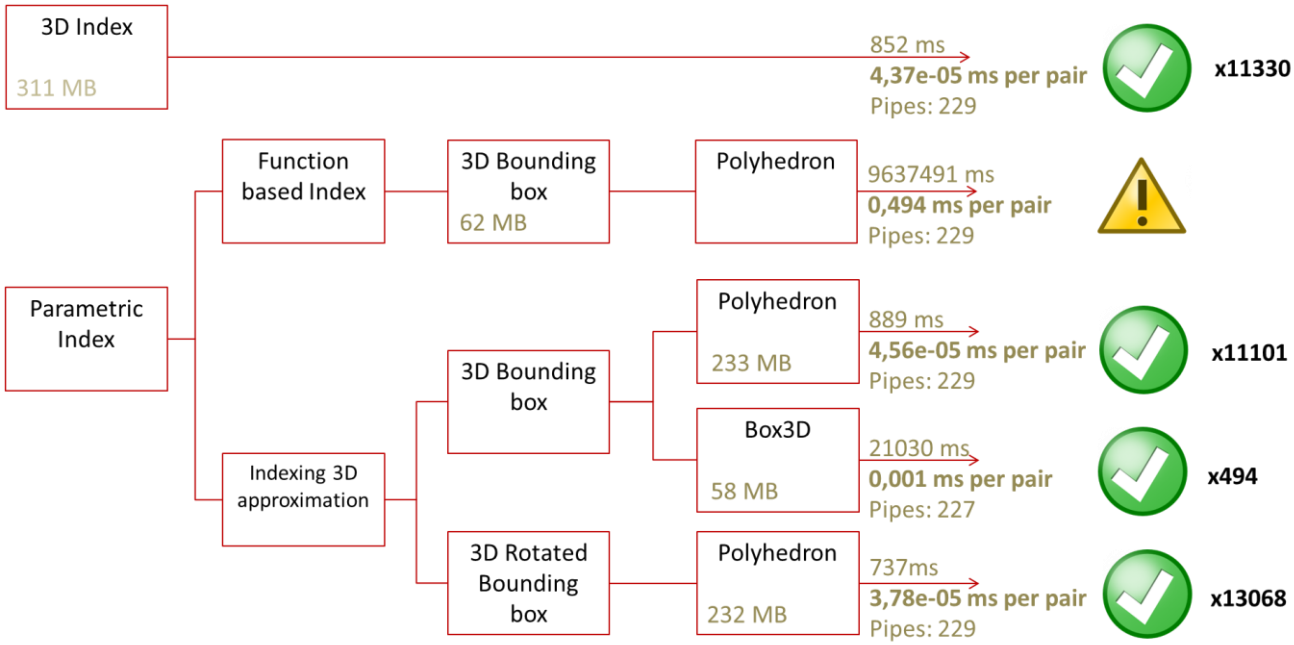


Figure 4.11: Explored methods for optimizing the 3D parametric DBMS performance, showing the query execution time, query execution time per pair, number of hits and optimization number in terms of query execution time.

[48] One is referred to Appendix B for the benchmark codes.

Table 4.5: Storage size of presented methods of section 4.2 versus 3D parametric methods of section 4.3.

		Table size (MB)	Index size (MB)	Total size (MB)
3D	3D non-parametric	295	16	311
3DP	3D parametric (function based index)	47	15	62
3DP2	3D parametric (polyhedron bounding box)	218	15	233
3DP3	3D parametric (Box3D bounding box)	58	0	58
3DRBB	3D parametric (rotated polyhedron bounding box)	218	14	232

Table 4.5 shows that saving a 3D approximation as a polyhedron (3DP2 & 3DRBB) requires a lot of space, which requires approximately the same storage as storing the geometry in 3D. However the query execution time is similar as a 3D non-parametric object, the main advantage of having a 3D parametric DBMS vanishes: Having low storage space. Keeping the low storage space in mind and looking at the query execution time, bots aspects are optimal when PostgreSQL's Box3D is saved. The query execution time drops from 5298,882 ms per pair to 0,001 ms per pair. The table storage also drops from 62 MB to 58 MB. This lower storage space is mainly because there is no index created for the Box3D data type.

Performing queries with the Box3D data type indeed lowers the query execution time, but the total hits of this geometry type in negatively affected. This bad result is caused by the fact that a Box3D data type is always casted to a 2D geometry for performing queries. This notion is also raised at the PostGIS Forum[49] (PostGIS, n.d.-a). However this task has been closed at the PostGIS Forum, it would be beneficial to fix this task with SFCGAL the extension. The SFCGAL extension allows to create polyhedra in PostgreSQL. These polyhedra can be used for casting the Box3D data type to a 3D polyhedral box. By fixing this task (casting the Box3D data type to a 3D polyhedron instead of a 2D polygon) the quality of performing queries with this geometry type will be as good as storing a 3D polyhedral bounding box.

The performance in terms of query execution time can be improved by creating an index, which allows Box3D indexing. This is possible by improving the spatial indexing code of PostGIS, and creating an index with this spatial index code. Due time limitation this will not be done during this graduation project.

[49] Raised notion has been closed (PostGIS, n.d.-a) and continues on PostGIS (n.d.-b). It is still an open issue for changing the Box3D casting to a 3D geometry. Both PostGIS (n.d.-a) and PostGIS (n.d.-b) are shown in Appendix C.

However the quality of the Box3D data type is not optimal yet, saving a Box3D bounding box in the 3D parametric table will be compared with the 3D non-parametric table in chapter 5. This geometry type has the most potency for choosing a 3D parametric DBMS above a 3D non-parametric DBMS, due its low storage size and improved query execution time. By saving PostgreSQL's Box3D the table design of the 3D parametric DBMS will be adapted according to figure 4.11. The new 3D parametric table is created with algorithm 4.17.

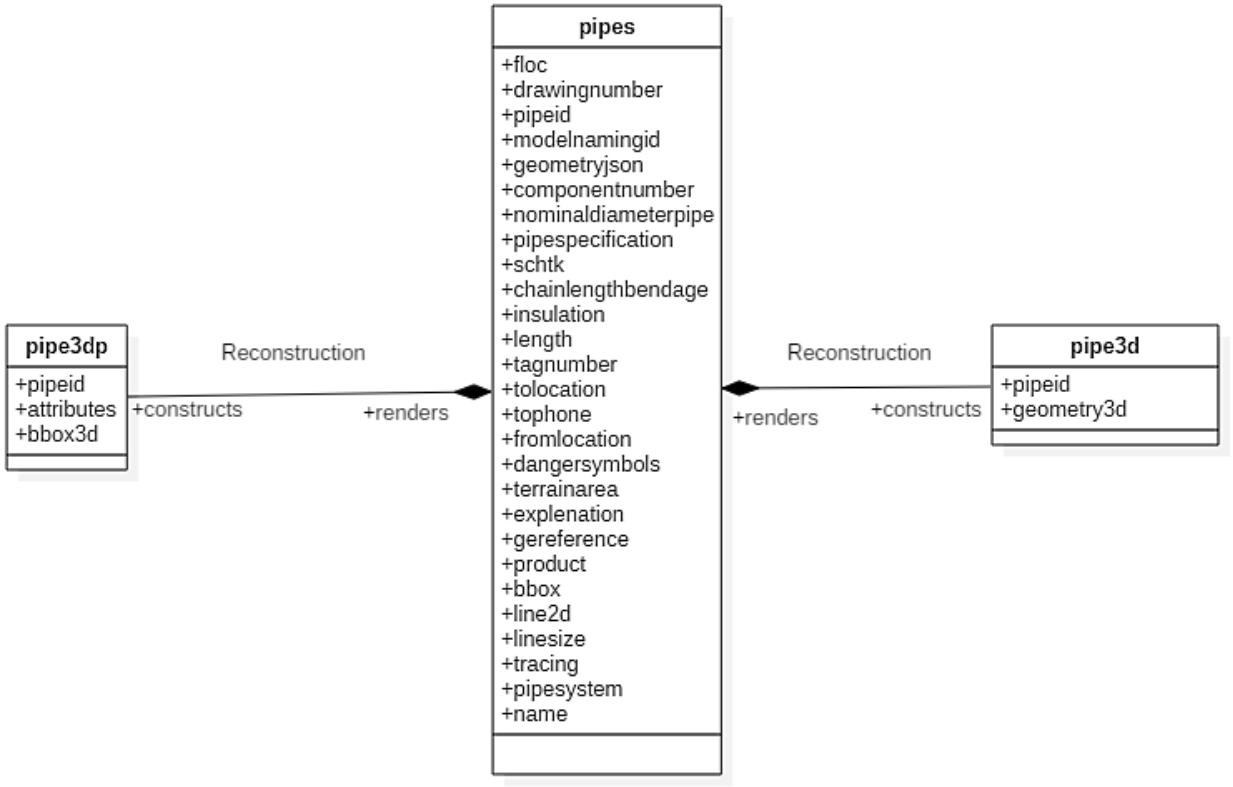


Figure 4.11: Changed DBMS design of the additional tables (pipe3d (storing the 3D solid polyhedra) and pipe3dp (parametric design)) for comparing the performance of both methods.

```
DROP TABLE IF EXISTS geo3d.pipe3dp;

CREATE TABLE geo3d.pipe3dp

(
   pipeid integer,
   attributes jsonb,
   bbox3d box3d,
   CONSTRAINT "FK_pipe3dp_pipes" FOREIGN KEY (pipeid)
         REFERENCES geo3d.pipes (pipeid) MATCH SIMPLE
         ON UPDATE NO ACTION ON DELETE NO ACTION
)
WITH (
   OIDS=FALSE
);
```

```
INSERT INTO geo3d.pipe3dp
SELECT *
FROM temptable
WHERE (attributes->'Type')::text='"Cylinder"';
```

Algorithm 4.17: Generating the new 3D parametric table of the pipes geometry table with only cylinder geometries. This 3D parametric table is referenced to the original pipes table.

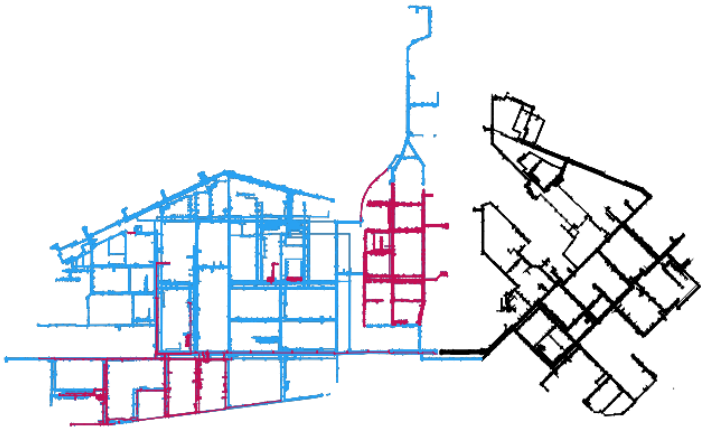
5

Scalability of the Data Set

This chapter compares the performance of having a 3D parametric DBMS versus a 3D non-parametric DBMS. This performance comparison is performed while increasing the data set from a mini to a full data set in section 5.1. In section 5.1 statistics are presented, ending with the result of this performance analysis in section 5.2.

5.1 Performance Analysis

The data set of Pernis has been chosen for testing the performance of a 3D parametric DBMS versus a 3D non-parametric DBMS. The data set of Pernis is the biggest data set of Fugro GeoServices and is for this reason the best data set to benchmark the scalability of having a 3D parametric DBMS versus a 3D non-parametric DBMS. The data set of Pernis consists out of 3 files, as shown in figure 5.1.



Legend: Data set file 1 94 MB, 84683 pipes
Data set file 2 252 MB, 202969 pipes
Data set file 3 193 MB, 137927 pipes

Figure 5.1: Chosen data set for benchmarking, location: Pernis. The data set consist of three files: red , blue and black.

With a program of Fugro GeoServices it is possible to split the files of the data set into smaller files. With help of this program data set file 2 and data set file 3 are split into 8 (file 2, blue) and 6 (file 3, black) files, resulting into 30 MB per file. File 1 (red) has not been split, because each insert round will insert 60MB at a time. This has resulted into 8 round for inserting the data set into the DBMS, with an average of 60 MB per round. After each insert the data set is benchmarked. For each insert new random points have been generated, as shown in figure 5.2.

In table 5.1 the storage size of both parametric and non-parametric table is compared with the native geometry table. Table 5.1 shows that the 3D non-parametric table has become more than twice as big, compared to the geometry table by only storing one geometry type.

Table 5.1: Storage size of the current geometry tables versus the additional 3D parametric - and 3D non-parametric table of the full data set.

	Table size (MB)	Index size (MB)	Total size (MB)
Pipes (current)	244	32	276
Pipe3dp (parametric)	126	0	126
Pipe3d (non-parametric)	639	32	671

5.1.1 Buffering

The buffering performance, based on increasing the data set (as shown in figure 5.2), is shown in table 5.2. The statistics (based on table 5.2) in terms of query execution time are presented in figures 5.3 (non-parametric) and figure 5.4 (parametric). As mentioned in section 4.2.2 the number of hits of the 3D parametric DBMS is negatively affected by the casting of PosgreSQL's Box3D to a 2D polygon.

Describing the function of the buffering performance in terms of the big O notation[50], figure 5.3 (non-parametric computation) has an *O(1)* – notation. This implies that the query execution time remains constant, without any increase in query execution time while increasing the data set. However the third point is seen as an outlier. Figure 5.4 (parametric computation) has an *O(n)* – notation, which implies that the query execution time has a steep grow while increasing the data set. This difference in the type of big O notation (figure 5.3 versus figure 5.4) is caused by the spatial index (figure 5.5). A spatial index optimizes the query execution time by giving directions where the approximately data is with help of branches, which results into an *O(1)* – notation.

[50] The big O notation is a notation to describe the functions behavior (Blatov, 2006).

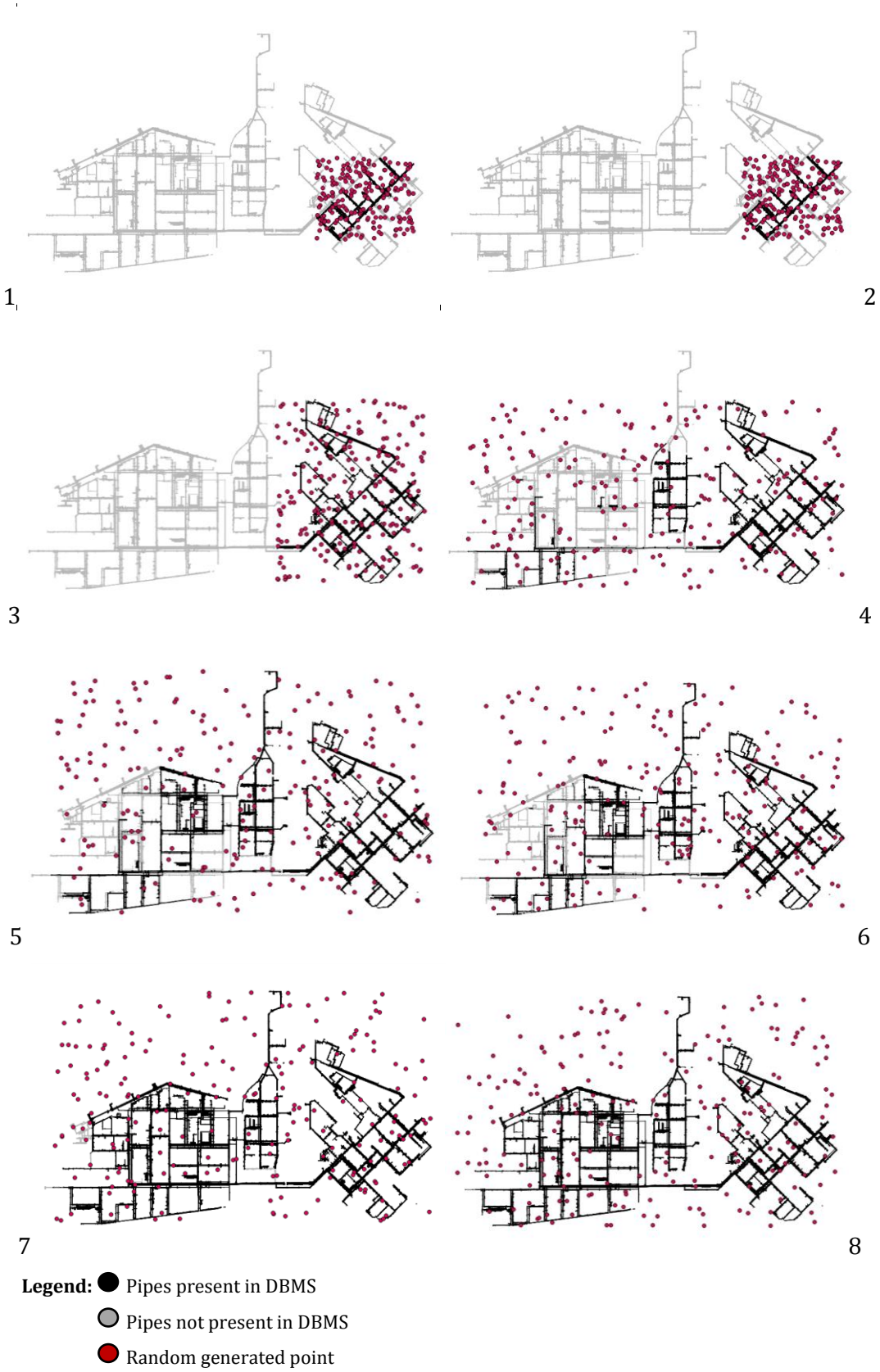


Figure 5.2: Random generated points (200 points in total per round) all over the data set, while increasing the data set from a mini- to a full data set (in 8 rounds).

Table 5.2: Performance of the buffering benchmark[51], 3D parametric (with stored Box3D)(3DP3) versus 3D non-parametric (3D) in terms of query execution time and number of hits.

Round		Amount of objects	Time (ms)	Amount of combinations*	Number of Hits	Number of pipes**	Time (ms) per pair***
1	3D	41471	114	8294200	613	556	1,38e-05
	3DP3	41471	10992	8294200	65	61	0,001
2	3D	93006	62	18601200	151	149	3,31e-06
	3DP3	93006	20773	18601200	42	42	0,001
3	3D	137927	174	27585400	613	569	6,31e-06
	3DP3	137927	32090	27585400	271	245	0,001
4	3D	222610	75	44522000	360	323	1,68e-06
	3DP3	222610	47261	44522000	177	144	0,001
5	3D	273283	42	54656600	102	102	7,60e-07,
	3DP3	273283	56691	54656600	23	23	0,001
6	3D	321192	96	64238400	419	296	1,50e-06
	3DP3	321192	67353	64238400	162	122	0,001
7	3D	374357	79	74871400	419	296	1,50e-06
	3DP3	374357	80897	74871400	162	122	0,001
8	3D	425579	80	85115800	882	768	9,34e-07
	3DP3	425579	90722	85115800	849	735	0,001

* Amount of geometries multiplied by the number of random points.
** Number of pipes: number of hits, distinguished on pipe id.
*** The time per pair is the computation time per true and false hits.

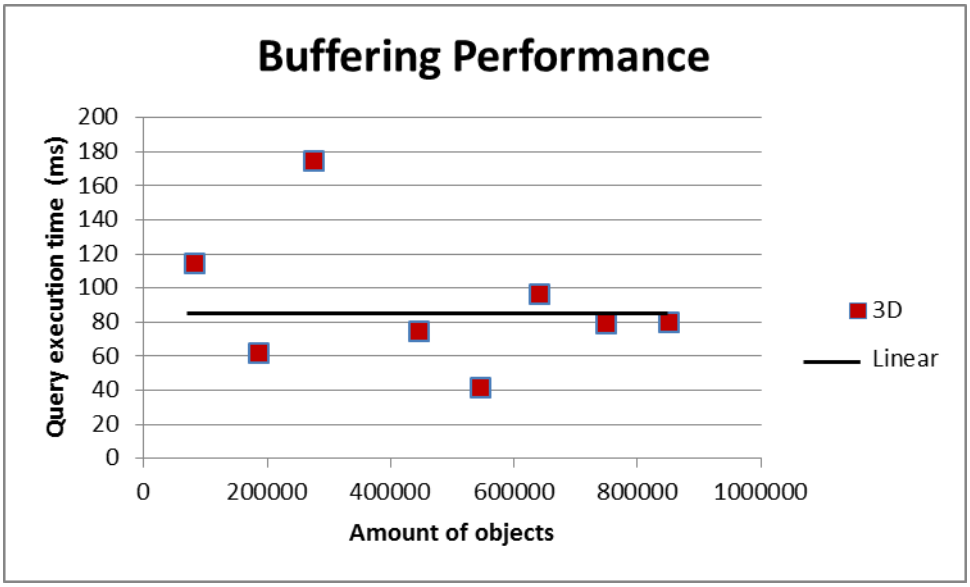


Figure 5.3: Statistical analysis of the buffering performance in 3D (non-parametric).

[51] One is referred to Appendix B for the benchmark codes.

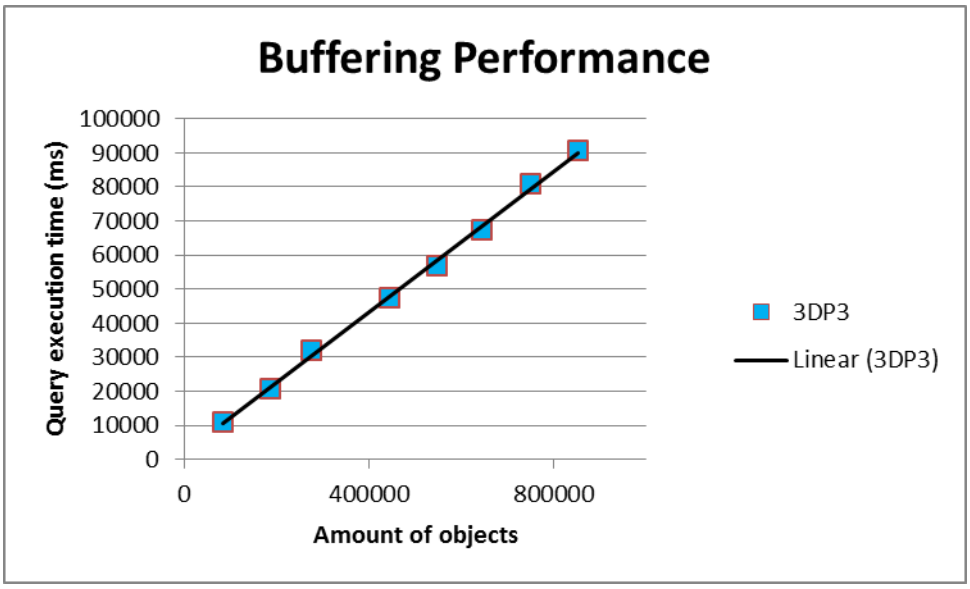


Figure 5.4: Statistical analysis of the buffering performance in 3DP3 (parametric, with stored Box3D).

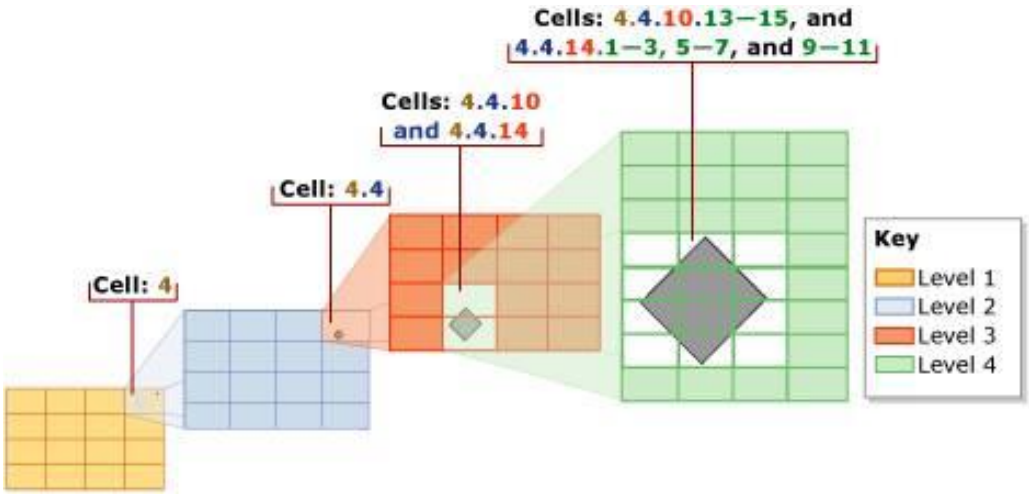


Figure 5.5: The usage of branches by spatial indexes, illustrated in 2D (Randal, 2008).

5.1.2 Intersections

The intersection performance, based on increasing the data set (as shown in figure 5.2), of the parametric computation versus the 3D non-parametric computation is shown in table 5.3. The statistics (based on table 5.3) in terms of query execution time is presented in figures 5.6 (non-parametric) and figure 5.7 (parametric).

Describing the function of the intersection performance in terms of the big O notation, figures 5.6 (non-parametric computation) and 5.7 (parametric computation) both have an *O(n)* – notation. This implies that the query execution time has a steep grow while increasing the data

set. Again around the 300.000 combinations an outlier is present in figure 5.6, which is also the case in figure 5.3 (around 300.000 objects).

Table 5.3: Performance of the intersection benchmark[52], 3D parametric (3DP3) versus 3D non-parametric (3D) in terms of query execution time and number of intersections.

Round		Time (ms)	Amount of combinations*	Number of intersections	Intersecting pipes**	Time (ms) per pair***
1	3D	183097	374976	198	138	0,488
	3DP3	253	4152	0	0	0,061
2	3D	42054	22646	28	17	1,857
	3DP3	113	1722	0	0	0,067
3	3D	149201	374874	228	138	0,389
	3DP3	4226	73008	0	0	0,058
4	3D	34172	129076	182	106	0,265
	3DP3	1782	31008	0	0	0,057
5	3D	17157	10302	20	14	1,665
	3DP3	45	506	0	0	0,090
6	3D	23367	174558	266	141	0,134
	3DP3	1508	25910	0	0	0,058
7	3D	109958	776590	428	249	0,142
	3DP3	41355	719500	0	0	0,057
8	3D	14998	133294	278	139	0,113
	3DP3	5595	97394	0	0	0,057

* Amount of combinations depends on the buffering benchmark. Only all pipes within the buffer in section 5.1.1 is used as input.
** Intersecting pipes: number of intersections, distinguished on pipe id.
*** The time per pair is the computation time per true and false hits.

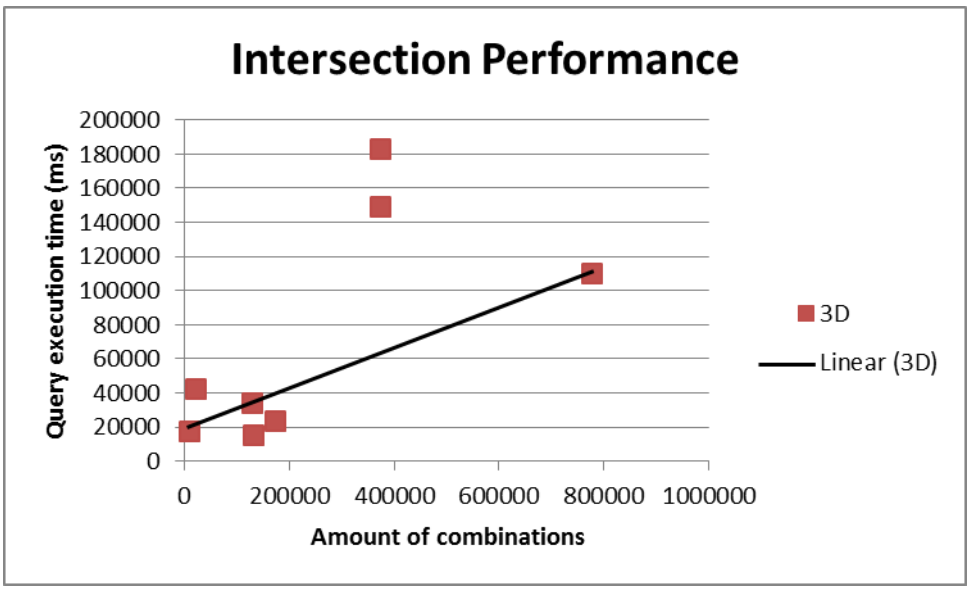


Figure 5.6: Statistical analysis of the intersection performance in 3D (non-parametric).

[52] One is referred to Appendix B for the benchmark codes

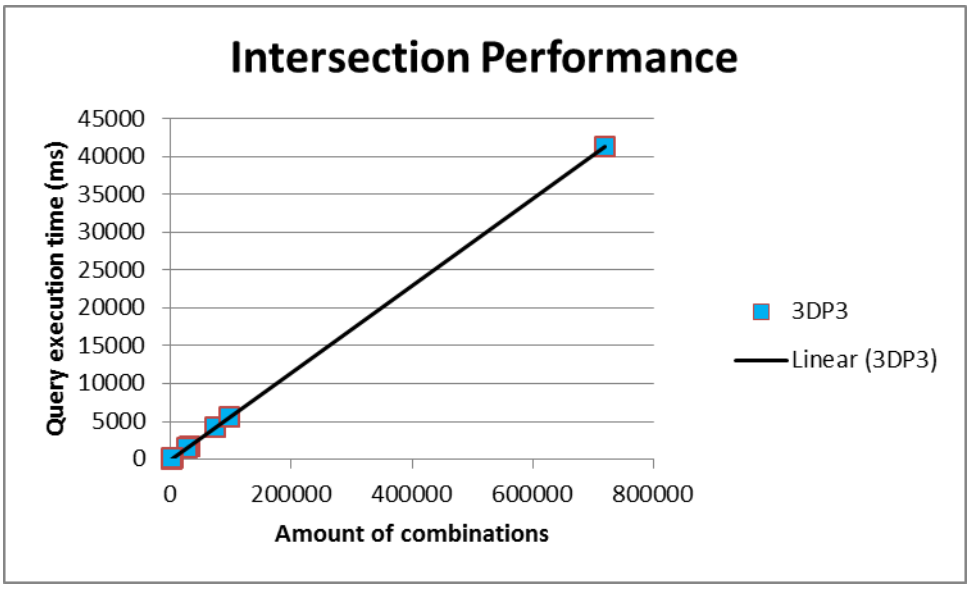


Figure 5.7: Statistical analysis of the intersection performance in 3DP3 (parametric, with stored Box3D).

Both the parametric buffering computation (figure 5.4) and parametric intersection computation (figure 5.7) have a similar big O notation, but this is not the case for the 3D non-parametric buffering computation (figure 5.3) and 3D non-parametric intersection computation (figure 5.6). Earlier the *O(1)* – notation has been explained by the spatial index in figure 5.3. The difference between the 3D buffering computation (figure 5.3) and 3D intersection computation (figure 5.6) is also caused by its spatial index. The effect of the spatial index is higher for the 3D buffering computations than for the 3D intersection computations, because the difference between whole data set versus part of the data set (3D buffering) and geometries within the part of the data set (3D intersection) is much bigger (figure 5.8).

5.1.3 Outlier Explanation

The present outliers in sections 5.1.1 and section 5.1.2 can be caused by two things: complex geometries or the diagonal pipes present within the data set. Looking at the data set and the errors the diagonal pipes seem to be the most logic explanation, because this graduation thesis has only incorporated cylinders. For this reason hypothesis 5.1 will be tested in this section.

Hypothesis 5.1: Diagonal geometries negatively affects the performance of a 3D non-parametric DBMS.

In order to validate hypothesis 5.1 the performance analysis has been performed two more times, as shown in figure 5.9. One benchmark is based on having minimal diagonal pipes and one benchmark is based on having a smaller bounding box of the data set with less empty spaces. Figures 5.10, 5.11, 5.12 and 5.13 show the statistical analysis of these two benchmarks.

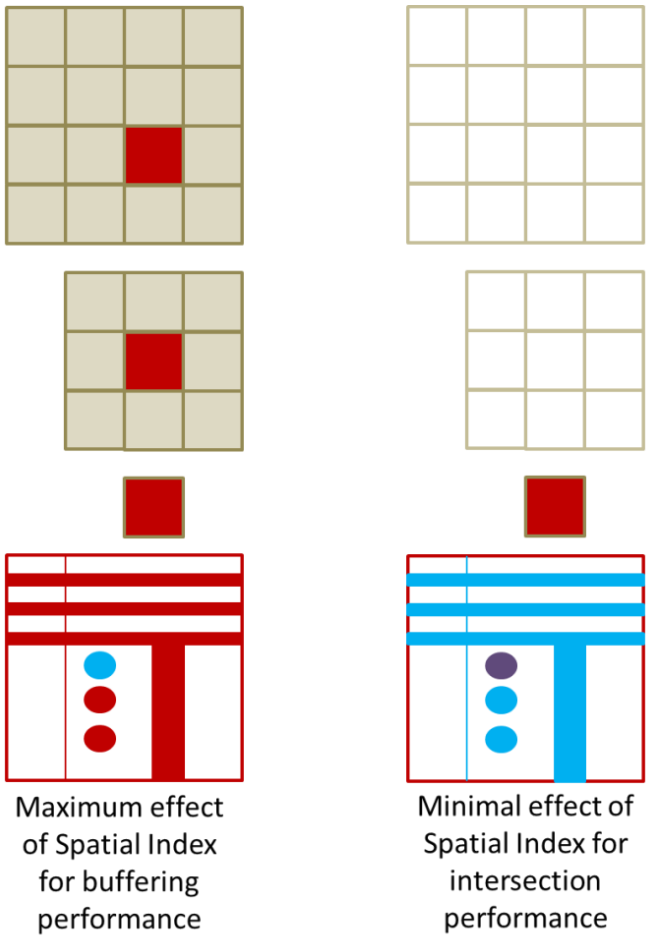


Figure 5.8: Maximum effect of having a spatial index (buffering performance) versus minimal effect of having a spatial index (intersection performance).

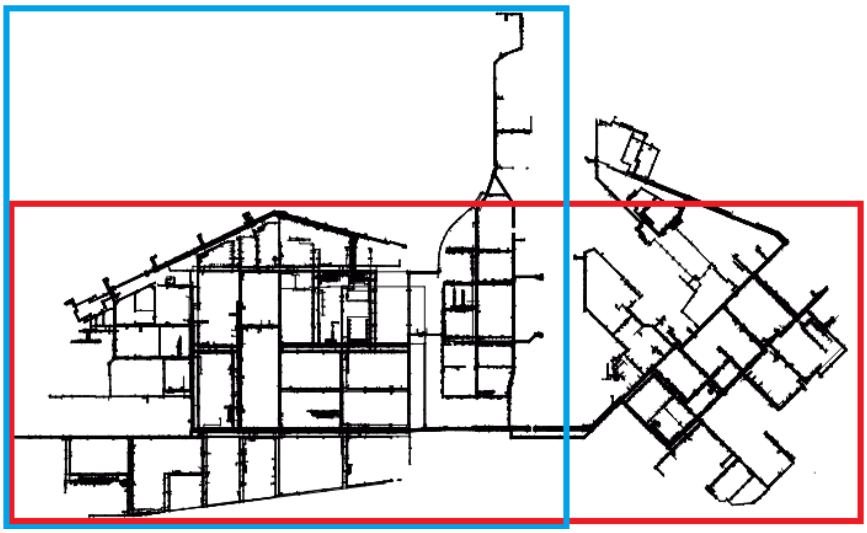


Figure 5.9: Hypothesis benchmark testing, comparing the effect of having minimal diagonal pipes (blue) with a smaller bounding box of the data set with less empty spaces (red).

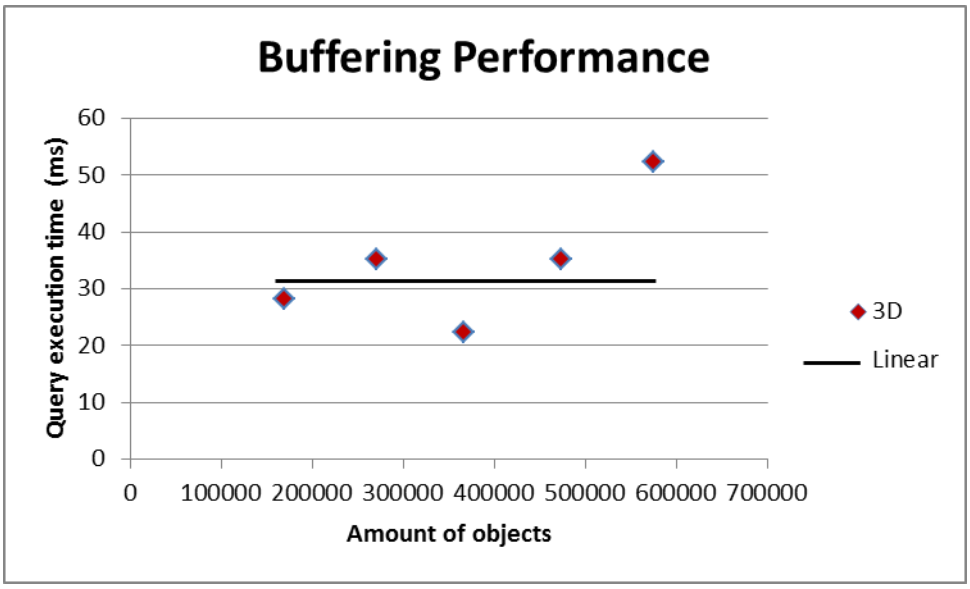


Figure 5.10: Statistical analysis of the buffering performance in 3D (non-parametric), having minimal diagonal pipes (blue in figure 5.9).

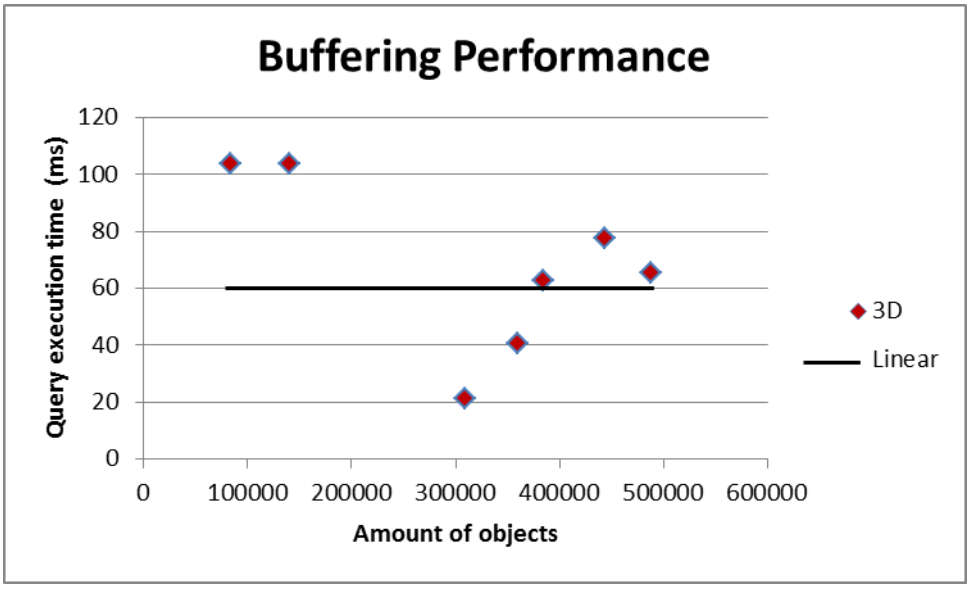


Figure 5.11: Statistical analysis of the buffering performance in 3D (non-parametric), having less empty spaces (red in figure 5.9).

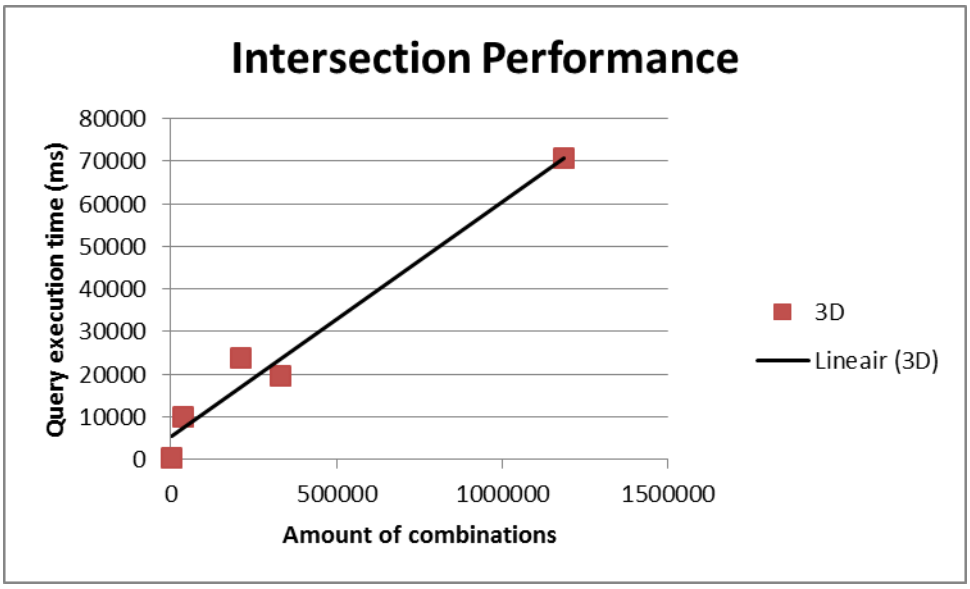


Figure 5.12: Statistical analysis of the intersection performance in 3D (non-parametric)), having minimal diagonal pipes (blue in figure 5.9).

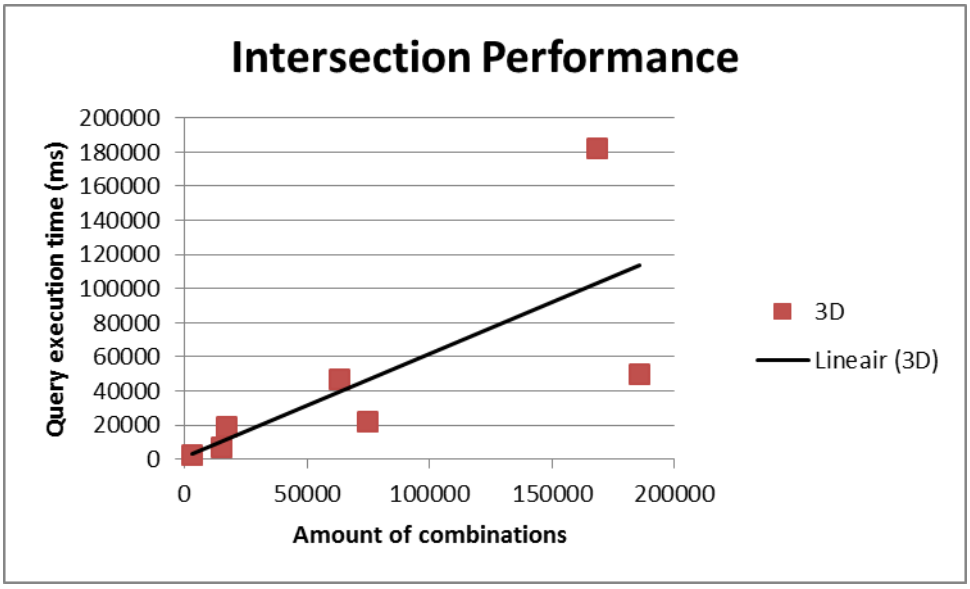


Figure 5.13: Statistical analysis of the intersection performance in 3D (non-parametric), having less empty spaces (red in figure 5.9).

Comparing figure 5.10 with figure 5.11 and figure 5.12 with figure 5.13 the benchmark having minimal diagonal pipes have a stable performance with some small deviations. The last point in figure 5.10 seems to be an outlier, but comparing this outlier with the outlier present in figure 5.3 the deviation is much smaller. For this reason the last point in figure 5.10 (with the largest deviation) is not regarded as an outlier. This means that the hypothesis 5.1 is true: Diagonal geometries negatively influence the performance, which is caused by the minimum bounding box (figure 5.14). This minimum bounding box is created by indexing the DBMS and has a larger error when the geometry does not aligns with the X-, Y-, or Z-axis, because the amount of candidate pairs for performing expensive calculations is much bigger. This notion is supported by figure 5.11: The benchmark begins with the diagonal pipes. Looking at figure 5.11 the first two benchmarks have a high query execution time compared to the other benchmarks present in figure 5.11, which is caused by having diagonal pipes. The first two benchmarks in figure 5.11 are based on file 3 (black) only. This file mostly contains diagonal pipes.

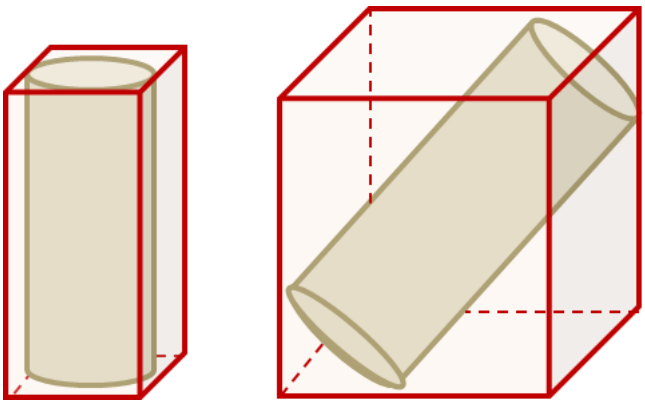


Figure 5.14: The minimum bounding box of an axis aligned geometry versus the minimum bounding box of a diagonal geometry (not axis aligned).

Hypothesis 5.1 is partially wrong: diagonal geometries does not only affect the performance of a 3D non-parametric DBMS. It can also effect a 3D parametric DBMS. By enabling an index on PostgreSQL's Box3D geometry and casting this Box3D to a 3D bounding box instead of a 2D polygon the performance will also be affected by the presence of diagonal geometries. This notion amplifies that hypothesis 5.1 should be adapted to hypothesis 5.2. From hypothesis 5.2 can be conducted that the performance of a DBMS can also be improved by aligning all geometries to the X-, Y- and Z-axis as much as possible.

Hypothesis 5.2: Diagonal geometries negatively affect the performance of a 3D DBMS, with 3D minimum bounding boxes as pre-filter.

5.2 Results

Looking at the performance analysis of section 5.1, it can be concluded that the 3D non-parametric table performs better than the 3D parametric table. Both query execution time and total hits are more accurate than the 3D parametric table, but the 3D non-parametric table also has a big disadvantage: the storage size.

From the statistical analysis of the buffering performance (figures 5.3 and 5.4) can be concluded that the 3D parametric DBMS has a steep grow in query execution time while enlarging the data set, which is caused by having no index. The 3D non-parametric table has a stable query execution time when enlarging the data set, which is mainly caused by its spatial index. A spatial index clearly optimizes the query execution time by giving a direction where the data approximately is. Optimization in terms of query execution time can also be retrieved by aligning the geometries to the X-, Y-, and Z-axis as much as possible.

Looking at the statistical analysis in figures 5.6 and 5.7 it is noticeable that both parametric intersection computation and PostgreSQL's intersection computation have a steep grow in query execution time while enlarging the data set. Table 5.3 shows that the parametric intersection computation is faster than the native PostgreSQL's intersection computation. The main clue for this optimization is in filtering the data by simple computations. The current parametric intersection computation only reconstructs the geometry in 3D when the shortest line between two geometries is intersecting with the starting and/or ending point of these two geometries.

Diagonal geometries negatively affects the performance of a 3D spatial DBMS when a minimum bounding box is used as pre filter. In order to optimize the performance of a 3D spatial DBMS with an index (based on a minimum bounding box) the data set has to be aligned as much as possible to the X- Y- and Z-axis.

6

Conclusion, Discussion and Recommendations

The difficulty in converting a 2D spatial DBMS to a 3D spatial DBMS has been the CAD-native data. Both worlds CAD and GIS have been developed separately, which has resulted into different standards and no unity. Many studies have been done in uniting both worlds, but these studies seldom provided a solution. For providing a solution for uniting both worlds this graduation project has conducted three research lines:

- Linking CAD and GIS;
- 3D reconstruction of CAD geometry in a 3D spatial DBMS;
- Comparing the functionality performance of a 3D parametric DBMS with a 3D non-parametric DBMS.

These three research lines give answer to the research questions of section 1.2.3 in section 6.1. With the obtained results in section 5.2 and answers to the research questions in section 6.1 it is possible to draw a conclusion in section 6.2, followed by a discussion in section 6.3. The recommendations for future directions are provided in section 6.4.

6.1 Answers to Research Questions

This section answers the research questions, which were presented in section 1.2.3.

Phase 0: Situation Understanding

1. What is the relationship between CAD and GIS?

   Section 2.1 shows that CAD and GIS are two technologies which are used in different phases in the life cycle of civil infrastructural projects (Hijazi, 2011), which has resulted into no unity in standards or interoperability. Section 2.5 shows that CAD has had a steady

evolution, which resulted into BIM. BIM incorporates some intelligent behavior of GIS (Karimi & Akinci, 2009).

2. What are the differences between CAD and GIS?

   Section 2.2 shows that CAD focusses more on the design (Karimi & Akinci, 2009) and shape of the object (Kazar et al., 2008), while GIS (section 2.3) is aiming for analysis by answering spatial oriented questions (Van Lanen et al., 1989).

3. What is a solid geometry?

   Section 3.1.2 defines a solid geometry as a boundary solid (representing the boundary between object and non-object, with a missing volume concept) or a true solid (solid with volume).

Phase 1: Data Inventory

4. What are the functional and technical requirements for Fugro's 3D application?

   Section 3.2 has set the following functional and technical requirements for Fugro's 3D application:

   - 3D query possibility;
   - Result must be visualized in 2D and 3D;
   - Volume calculation possibility;
   - Buffer calculation (of a certain area);
   - Spatial relationship calculation;
   - Length calculation;
   - Distinguish properties of the components, e.g. current state;
   - Topology determination (no high priority).

5. What data does the data set of the petrochemical industry consist of?

   Section 3.2.3 shows that the data set of the petrochemical industry consist out of parametric combination models and is built with help of the following geometries (section 3.2.2): cone, cylinder, dome, elbow, solid sphere.

Phase 2: Potential Exploring

6. Which potential methods exist for the storage of solid CAD geometries in DBMSs?

   Section 3.1 shows that parametric design and solid polyhedral modelling enables the storage of solid CAD geometries in PostgreSQL. Parametric design results into a 3D parametric DBMS and solid polyhedral modelling results into a 3D non-parametric DBMS (section 3.2.4).

Phase 3: 3D data storage

7. What are the advantages and disadvantages of the potential methods for storing the solid geometry ?

   Section 5.1 shows that having a 3D non-parametric DBMS results into a high storage space of the DBMS, but has a low query execution time. In the current implementation a 3D parametric DBMS results into low storage space, but has a much higher query execution time compared to a 3D non-parametric DBMS. The high query execution time of the 3D parametric DBMS should be improved. In terms of attribute - and spatial relationship calculation it is cheaper to calculate attributes with 3D parametric objects, than 3D non-parametric objects. Both 3D parametric and non-parametric enables a bi-directional flow between DBMSs and GIS applications.

8. Which method is the most efficient method for storing solid CAD geometries of the petrochemical industry in a spatial DBMS?

   Section 5.2 shows that the most efficient method for storing solid CAD geometries in terms of query execution time is the 3D non-parametric DBMS. In terms of storage size the 3D parametric DBMS is the most efficient method for storing solid CAD geometries.

9. How is it possible to store the data and incorporate the constrains of Fugro's 3D application?

   Section 3.2.4 shows that the geometries, both parametric and non-parametric, can be stored in the DBMS by aggregating a 3D table to the existing geometry table. With help of this additional table it is possible to query the data set in 3D, as set by the functional and technical requirements.

Phase 4: Performance optimization

10. How can the performance of the 3D DBMS for solid CAD geometries be optimized?

    The performance of a 3D DBMS can be optimized with help of a spatial GIST index. A non-spatial DBMS (3D parametric DBMS) requires a function based index or a stored 3D approximation of the geometry for indexing the DBMS. Section 4.3 has proven that indexing a 3D approximation of a geometry is more efficient than a function based index as spatial

index (section 4.3). Performance optimization can also be retrieved by rotating the data set in such way, that most of the geometries align with the X-, Y- and Z-axis (section 5.1.3).

Phase 5: Testing & evaluation

11. How does the 3D DBMS fulfill the requirements of Fugro GeoServices?

Looking at the technical and function requirements, set by Fugro GeoServices in section 3.1, both 3D parametric and non-parametric DBMSs fulfill the requirements, with exception of the topology determination (which had no high priority) and volume calculation (only in case of a 3D non-parametric DBMS). However the volume calculation can easily be implemented parametrically as done for the area calculation in section 4.2.

Phase A: Visualization

12. How can the queried solids be visualized in 3D?

Section 4.1 shows that the queried solids can be visualized in X3D, by converting the geometries to X3D geometries in PostgreSQL. This geometry conversion is 'built in' in PostgreSQL. With help of these converted X3D geometries it is possible to generate a X3D-file.

6.2 Conclusion

The main research question of this graduation project has been (section 1.2.2):

How is it possible to store and visualize solid geometries in a spatial DBMS suitable for the petrochemical industry?

With help of PostgreSQL it is possible to visualize solid geometries with a spatial DBMS by converting the stored 3D geometries to X3D geometries with a 'built-in' conversion. With these X3D geometries a X3D-document can be generated (section 4.1) for visualization purposes. For storing a parametric combination model (CSG) as a solid geometry in a 3D spatial DBMS the geometry must be reconstructed as a solid polyhedron with the constrains of Kazar et al. (2008) and Ledoux (2013) (section 3.1). The reconstucted geometries can be stored as a JSON (containing the geometry attributes) with a 3D bounding box (PostgreSQL's Box3D), allowing to have a 3D parametric DBMS. The reconstructed geometries can also be stored as a polyhedron (which is an approximation of the exact geometry with a maximum deviation of 2,4 cm), allowing to have a 3D non-parametric DBMS (section 3.2.4). This DBMS design enables a bi-directional flow between DBMSs and GIS applicatios for analyses (figure 6.1).

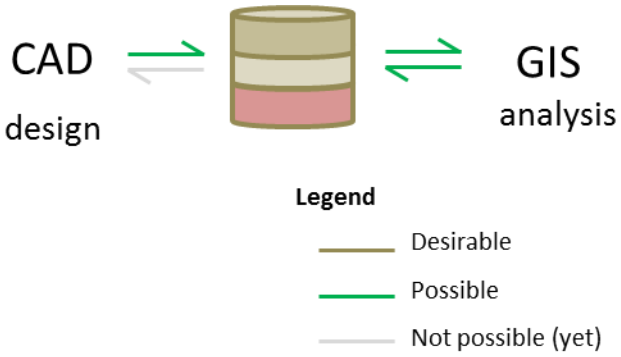


Figure 6.1: Achieved bi-directional flow between DBMSs and GIS applications for analyses. This figure also includes the current one directional flow between CAD applications and DBMSs, achieved by Fugro GeoServices.

6.3 Discussion

When comparing a 3D parametric DBMS with a 3D non-parametric DBMS it is noticeable that a 3D non-parametric DBMS is ready to implement with a good performance and the 3D geometry is easy accessible for web applications (without needing a 3D reconstruction by the web application). However some functions do not work well (e.g. the area calculation of PostgreSQL) or are not implemented (e.g. volume calculation). The big storage space of the 3D non-parametric DBMS is a tradeoff. In order to minimize the storage space it is advisable to have a 3D parametric DBMS instead of a 3D non-parametric DBMS. For this option some effort needs to be done:

- For every geometry type a function has to be written for 3D reconstruction and generating a 3D bounding box (as a Box3D data type);
- For every 3D spatial relationship a parametric function has to be written, minimizing the 3D reconstruction of a geometry. This parametric function must also incorporate relationship computations between different geometry types (e.g. cylinder – sphere);
- An index has to be made for enabling indices on PostgreSQL's Box3D;
- PostgreSQL's Box3D must be casted to a 3D polyhedral bounding box instead of a 2D polygon.

The query execution time of the 3D parametric DBMS for the test data set takes roughly 1000x more time for performing queries (in terms of buffering). It must be stressed out that this performance is based on having no index. Enabling the ability to create a spatial index on the stored Box3D will benefit and stabilize the query execution time. Now, by having no index, every row is examined sequentially by the query condition. By having a spatial index not every row

has to be examined, because the index is queried by branches. This will result into a similar stabilized query execution time as having a 3D non-parametric DBMS.

6.4 Recommendations

Where this graduation project ends there are also some notions left open, which has to be addressed:

- Adapting the current Pipe JSON format of Fugro GeoServices.
  During this graduation project an adapted version of Fugro's Pipe JSON has been used to simplify the query ability of the DBMS. For the makers of Fugro's Pipe JSON it is recommended to adapt the format. The current Pipe JSON is stored as a text string in the DBMS. This graduation project has changed the storage to a binary JSON format. Figure 3.8 and figure 3.9 show the difference between both formats in section 3.2.3.

- Reconstructing all geometry types of Fugro GeoServices.
  Due time limitation it has not been possible to reconstruct all geometry types of Fugro GeoServices, such as domes, spheres, elbows, etc. It is desirable to reconstruct all geometries for translating the current 2D spatial DBMS to a 3D spatial DBMS.

- Making 3D spatial relationship functions, based on parametrical computations.
  This graduation project has proved that spatial relationships can be computed quicker parametrically than the native PostgreSQL function, by performing cheap pre-computations. These parametric spatial relationship functions must be able to compute the spatial relationship between different geometry types. For this reason it is important to make a library of functions for computing the spatial relationship for each possible combination of geometry types.

- Making functions for computing the attributes of the objects, such as area and volume.
  With help of the stored attributes of the geometry it is possible to calculate the attributes of the objects parametrically in an efficient way, such as area and volume. Section 4.2.4 has proven that it is more efficient and accurate to compute attributes of spatial objects parametrically. It also could be possible to make functions for computing the attributes of objects with the geometry as input, but this computation will not be so efficient compared to a parametric computation (with help of the parametric attributes of the objects) in terms of query execution time.

- Improving the area computation in 3D of PostgreSQL (for polyhedrons).

  During this graduation project it was desired to make a comparison between the 3D area computation of PostgreSQL and a parametric area computation of the spatial object. This comparison could not be done, because the implemented function for computing the area of an object in 3D is not well implemented by PostgreSQL for polyhedra.

- Adapting the geometry casting of PostgreSQL's Box3D to a 3D polyhedron.

  PostgreSQL's Box3d data type casts the Box3D as a 2D polygon for performing computations. This negatively influences the accuracy of the performance. With help of the SFCGAL extension this notion can be solved by converting the Box3D data type to a 3D polyhedron instead of a 2D polygon.

- Building an index for PostgreSQL Box3D data type.

  PostgreSQL does not support indices on the Box3D data type. It is possible to index this geometry type, by searching for the indexing source code in PostGIS. Creating this index will have a positive effect on the performance of the 3D parametric DBMS.

- Store the topology of the data set of Fugro GeoServices.

  Storing the topology of the data set of Fugro GeoServices will enable to follow the flow of the geometries stored in the DBMS. This flow enables spatial relationship computations with help of tables instead of geometries. This method for computing spatial relationships saves computation time (Brugman, 2010). The storage of topology will also enable to reduce the amount of points by aggregating the starting and ending point of two geometries, which will lead to less storage space of the DBMS and a performance optimization of the WebGL clients (Guerrero Iñiguez, 2012) (Fugro's web application is based on WebGL).

- Research in a bi-directional flow between CAD and DBMSs.

  Now when a bi-directional flow has been achieved between DBMSs and GIS applications for analyses it is desirable to have a bi-directional flow between CAD applications in order to unite the worlds of CAD and GIS. This bi-directional flow between CAD applications and DBMSs must allow direct data storage in - and direct data retrieval from the DBMS. This will enable up-to-date data, which is easy updated in the DBMS.

- Research in adapting the DBMS structure of Fugro GeoServices.

  Fugro GeoServices is dividing the data set into geometry classes (architectural, pipes, etc.). It could be beneficial to adapt the structure of the DBMS, by dividing the geometry classes into parent-child relationships.

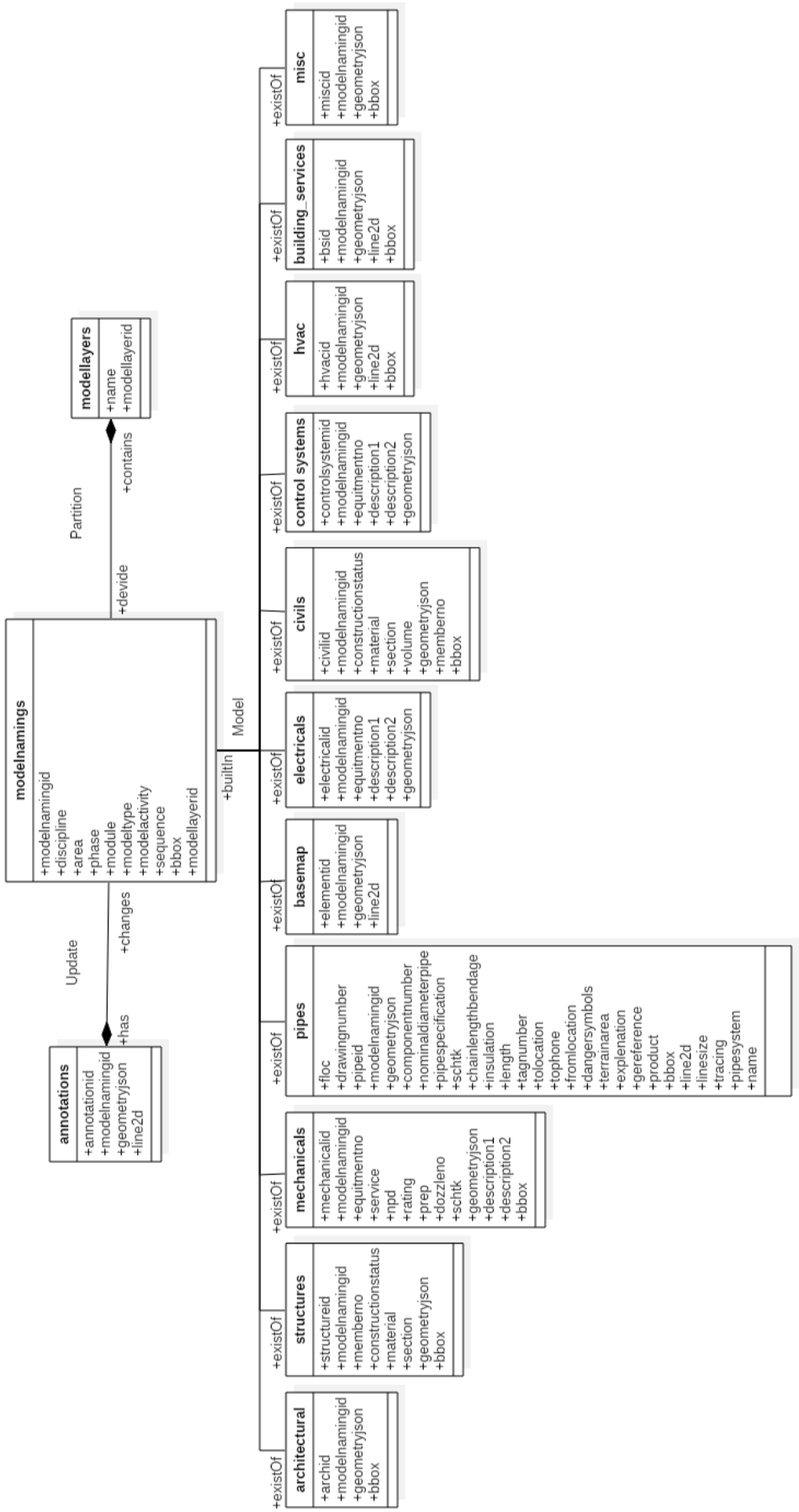
- Research in optimizing the minimum bounding boxes of diagonal geometries.

  This graduation project has shown that diagonal geometries, which do not align with the X-, Y- and Z-axis negatively influence the performance (section 5.1.3). This is caused by the minimum bounding boxes of diagonal geometries. Research is needed how to detect diagonal geometries for creating a more efficient minimum bounding box which contains less error, such as a rotated bounding box. It could be beneficial to create an index type which uses rotated bounding boxes as pre-filter instead of a minimum bounding box, but this will increase the storage space of the index: A minimum bounding box can be stored with two or four points, while a rotated bounding box must be stored with eight points. More research is needed to investigate the costs and benefits of having a rotated bounding box in terms of query execution time improvement and storage size.

A

Appendix: UML Diagram

UML Diagram Current DBMS



B

Appendix: Thesis Functions and Codes

For the functions and codes of this graduation project one is referred to the following website: https://github.com/mramkisoen/solidgeometry

Table B.1 is provided as guideline where to find the functions and codes at the website.

Table B.1: Guideline for codes on GitHub Website.

Chapter	Section	Code/Function	Location
3	3.2.3	Pipe JSON conversion code	PythonCodes
4	4.1	__ST_3DCylinder	plSQLfunctions
4	4.1	__ST_DotProd	plSQLfunctions
4	4.1	ST_CircelPoly	plSQLfunctions
4	4.1	ST_3DReconstruct	plSQLfunctions
4	4.1	X3D document SQL Code	SQLcodes
4	4.2.1	Mini-Benchark Coding	Benchmark/BM1
4	4.2.1	randompointsinpolygon3d	plSQLfunctions
4	4.2.1	ST_bboxline	plSQLfunctions
4	4.2.3	ST_ParaIntersect	plSQLfunctions
4	4.2.4	ST_CylinderArea	plSQLfunctions
4	4.3	Mini Benchmark coding stored 3D approximation	Benchmark/BM3
4	4.3	ST_3DCylinderbb	plSQLfunctions
4	4.3	ST_bbox	plSQLfunctions
4	4.3	ST_CircelPolybb	plSQLfunctions
4	4.3	ST_Rbb	plSQLfunctions
5	5.1.1	Benchmark coding data upscale	Benchmark/BM4
5	5.1.2	Benchmark coding data upscale	Benchmark/BM4

C

Appendix: PostGIS Forum

PostGIS forum (PostGIS, n.d.-a):

#103 closed task (wontfix)

Opened 7 jaar ago
Closed 4 jaar ago

Box3D always gets cast to 2D geometry

Gemeld door:	robe	Eigenaar:	strk
Prioriteit:	medium	Mijlpaal:	~~PostGIS 2.0.0~~
Component:	postgis	Versie:	trunk
Sleutelwoord:	boxes	Cc:	

Beschrijving (laatst gewijzigd door strk) Δ

Actually not sure if this is solvable. But ST_SetSRID, ST_Force_3DZ all lose the Z coordinate value of a BOX3D.

Example:

```
SELECT ST_AsEWKT(ST_Force_3DZ(ST_Extent3D(foo.the_geom))) As b3extentpoly,
ST_Extent3D(foo.the_geom) As b3extent FROM (SELECT ST_MakePoint(x,y,z) As the_geom
    FROM generate_series(1,3) As x
        CROSS JOIN generate_series(1,2) As y CROSS JOIN generate_series(0,2) As Z) As foo;
```

Yields: b3extentpoly b3extent POLYGON((1 1 0,1 2 0,3 2 0,3 1 0,1 1 0)) ;BOX3D(1 1 0,3 2 2)

I would expect the POLYGON value to have some 2 z coords in there.

▼ **Wijzigingenoverzicht** (8)

Oldest first Newest first
Comments only

Changed 7 jaar ago by mcayland — comment:1

The issue here is that the cast from BOX3D to geometry only works in 2 dimensions. Unfortunately the behaviour of the 3rd dimension is not really defined when casting from a cube, so we could interpolate the Z coordinates but at the end of the day that doesn't really help you at all.

Since there is no well-defined behaviour, we may as well stick with the 2D conversion with the Z coordinate set to 0 and mark this as 'Won't fix'.

ATB,

Mark.

Changed 5 jaar ago by strk — comment:2

- **Beschrijving** gewijzigd (diff)
- **Mijlpaal** ingesteld op *PostGIS 2.0.0*
- **Oplossing** *wontfix* verwijderd
- **Prioriteit** veranderd van *low* naar *medium*
- **Status** veranderd van *closed* naar *reopened*
- **Versie** ingesteld op *trunk*

Wait a sec, box3d is not a cube, is it ? It's defined by 2 points only, you can't define a cube like that.

I've hit this problem as well, while trying to get a 3d envelope of a 2.5d geom.

Changed 5 jaar ago by strk — comment:3

- **Beschrijving** gewijzigd (diff)
- **Eigenaar** veranderd van *robe* naar *strk*
- **Status** veranderd van *reopened* naar *new*

Changed 5 jaar ago by robe — comment:4

strk, Can you ever get a valid 2.5D polygon out of a box3d unless the zs of both points are equal? The box3d in essence defines something that can not normally exist in 2.5D. It's a 2d box accompanied by a Z yardstick.

Changed 5 jaar ago by strk — comment:5

uhm. now that's a good point. So, what's the plan for 3d indices and such ? Which gets back to... shouldnt' we define operators of boxes rather than geometries ?

The only reason why I hit this issue was for 2.5d topology, in particular the mbr field of face table. I guess I can get away by keeping it 2d no matter topology vertex dimensionality, but wouldn't want to go against the wind if gbox is introducing smarter boxes.

Changed 5 jaar ago by strk — comment:6

To have a shared view on this.. BOX3D actually defines a cube, in terms of min-max range in the X/Y and Z dimension.

You _might_ represent a cube with polyhedralsurface I guess.

Changed 4 jaar ago by strk comment:7

- **Sleutelwoord** *boxes* toegevoegd

Changed 4 jaar ago by strk comment:8

- **Oplossing** ingesteld op *wontfix*
- **Status** veranderd van *new* naar *closed*

Ticket #1329 obsoletes this one, as the only reason why you'd want to cast a BOX to a GEOMETRY is to set a srid for use with operators.

Noot: Zie TracTickets voor hulp bij het gebruik van tickets.

PostGIS forum (PostGIS, n.d.-b):

#1329 new enhancement

Opened 4 jaar ago
Last modified 8 maanden ago

Change BOX3D to be SRID and dimension aware

Gemeld door:	robe	Eigenaar:	pramsey
Prioriteit:	medium	Mijlpaal:	PostGIS Future
Component:	postgis	Versie:	trunk
Sleutelwoord:	box	Cc:	

Beschrijving (laatst gewijzigd door robe) Δ

Discussion here

http://www.postgis.org/pipermail/postgis-devel/2011-November/016216.html

To be followed up after float to double ~~#1328~~.

As a consequence of this change, the following also needs to be done for backward compatibility.

box3d_extent will be deprecated and possibly removed, though I think that might break some apps removing it. I have one to test that theory.

deprecate box2d (possibly get rid of it). I think there is too much code out there that relies on box2d that getting rid of it is a pipe dream the moment. So we might want to deprecate it now, remove it from the docs, and encourage people to not use it in a non-violent demonstration of disapproval.

1) ST_Extent returns box3d just like ST_3DExtent currently does. 2) Both ST_3DExtent and ST_Extent be changed to strip the SRID of the improved box3d type by default, with an optional argument -- keep_srid (defaulted to false) for those like strk who really want it and want to override the behavior to make it true.

All this stuff better not push out the release time beyond our slated February 2012 or I'll be steaming mad (at least mad for a week).

▼ **Wijzigingenoverzicht** (8)

Oldest first Newest first
Comments only

Changed 4 jaar ago by robe comment:1

- **Beschrijving** gewijzigd (diff)

Changed 4 jaar ago by strk comment:2

See ~~#103~~ for a common use case (setSRID on box3d dropping Z)

Changed 4 jaar ago by strk comment:3

If it's just for me that we have to make this change I'll need to double check my priorities :)

Seriously my whole idea here was to simplify BOXes for use with BOX oriented operations, since we have a few. Mostly extent and operators and indices.

The idea is to make it transparent for client code that goes:

```
CREATE INDEX mytable_spatial_index on mytable (<somecolumn>);
SELECT * FROM mytable WHERE <somecolumn> && ST_MakeEnvelope(...);
SELECT * FROM mytable WHERE <somecolumn> && ST_SetSRID'BOX(...)'; -- legacy
```

The above with <somecolumn> possibly being of type GEOMETRY or GEOGRAPHY or RASTER or TOPOGEOMETRY.

Did anyone already research on this topic and has conclusions about it ? Or I'd spend some time on it.

Changed 4 jaar ago by strk comment:4

- **Mijlpaal** veranderd van *PostGIS 2.0.0* naar *PostGIS 2.1.0*
- **Prioriteit** veranderd van *blocker* naar *medium*

We're too close to a release to do anything like this. The whole box/envelope discussion needs to be postponed to 2.1, or even future.

Changed 3 jaar ago by strk comment:5

- **Sleutelwoord** *box* toegevoegd

So back to this issue. I'd want a real "box" type which can hold SRID and any number of dimensions. It can be float or double I don't really care at this very moment, but it's a way to easily handle a rectangle, get its width and height, grow it or shrink it, expand to include a point or another box etc...

It could be the only type on which indices and bounding-box based operators are defined and would have an implicit cast from all types (raster, geometry, geography, TopoGeometry) to take advantage of those indices and operators.

A good next step will likely be a wiki page, to link here.

Changed 3 jaar ago by robe comment:6

- **Mijlpaal** veranderd van *PostGIS 2.1.0* naar *PostGIS Future*

I don't think this is going to happen anytime soon. punt.

Changed 8 maanden ago by strk comment:7

Stumbled upon this one while thinking I needed a BOXND to find extent of an 4D table... To be honest we could get away with all these boxes by using a two-points LINESTRING. It would embed SRID and dimension flags.

Only it would be slightly larger in size than a simple box, and have less semantic associated with it. How would you feel about an ST_NDExtent() returning geometry ? Doesn't it sound too loose ?

Changed 8 maanden ago by strk comment:8

Related, PostgreSQL 9.2 introduced "range types":
http://www.postgresql.org/docs/9.2/static/rangetypes.html

Boxes in any dimensions could be encoded as array of ranges. There would be no SRID awareness though...

D

Appendix: Project Planning

This graduation project has been divided into 5 phases (P1-P5) and is planned to monitor the overall progress. The important dates and deadlines for this graduation thesis are shown in table C.1.

Table C.1: Important dates of this graduation project.

	Week		Date	Description
P1	17	4.1	April 16th, 2015	Presentation Progress review Graduation plan
	24	4.8		<1 week before P2> Submit final graduation plan to all
P2	25	4.9	June 19th, 2015	Presentation Go / no go assessment Graduation plan
	33	5.6		<1 week before P3> Submit draft thesis to all
P3	34	5.7	August 28th, 2015	Presentation Colloquium midterm
	36	1.1	September 1st , 2015	Final Application date P4 Go/No Go
	39	1.4		<1 week before P4> Submit draft thesis to all
P4	40	1.5	September 28th, 2015	Presentation Go / no go assessment Process
	40	1.5	October 2nd, 2015	Final Application date P5 Go/No Go
	44	1.8-1.9		<1 week before P5> Submit thesis to all
P5	45	1.9-1.10	November 2nd, 2015	Final Public Presentation & assessment

Table C.1 has been set as a framework to plan this graduation project. This graduation project has been planned in an iterative lifecycle approach, with 3 obligatory iterations. The Gantt planning for this graduation project is shown in figure C.1.

Looking at the planning of this graduation project in figure C.1 it is visible that there has been a delay during the P3 period. The main reason for this delay has been the estimation of the length of the first iteration. Reconstructing the 3D geometry properly took more time than expected. When this 3D geometry has been reconstructed properly the iterative life cycle came into force, by investigating the clients (Fugro GeoServices) requirements. This graduation project succeeded in making 3 iterations.

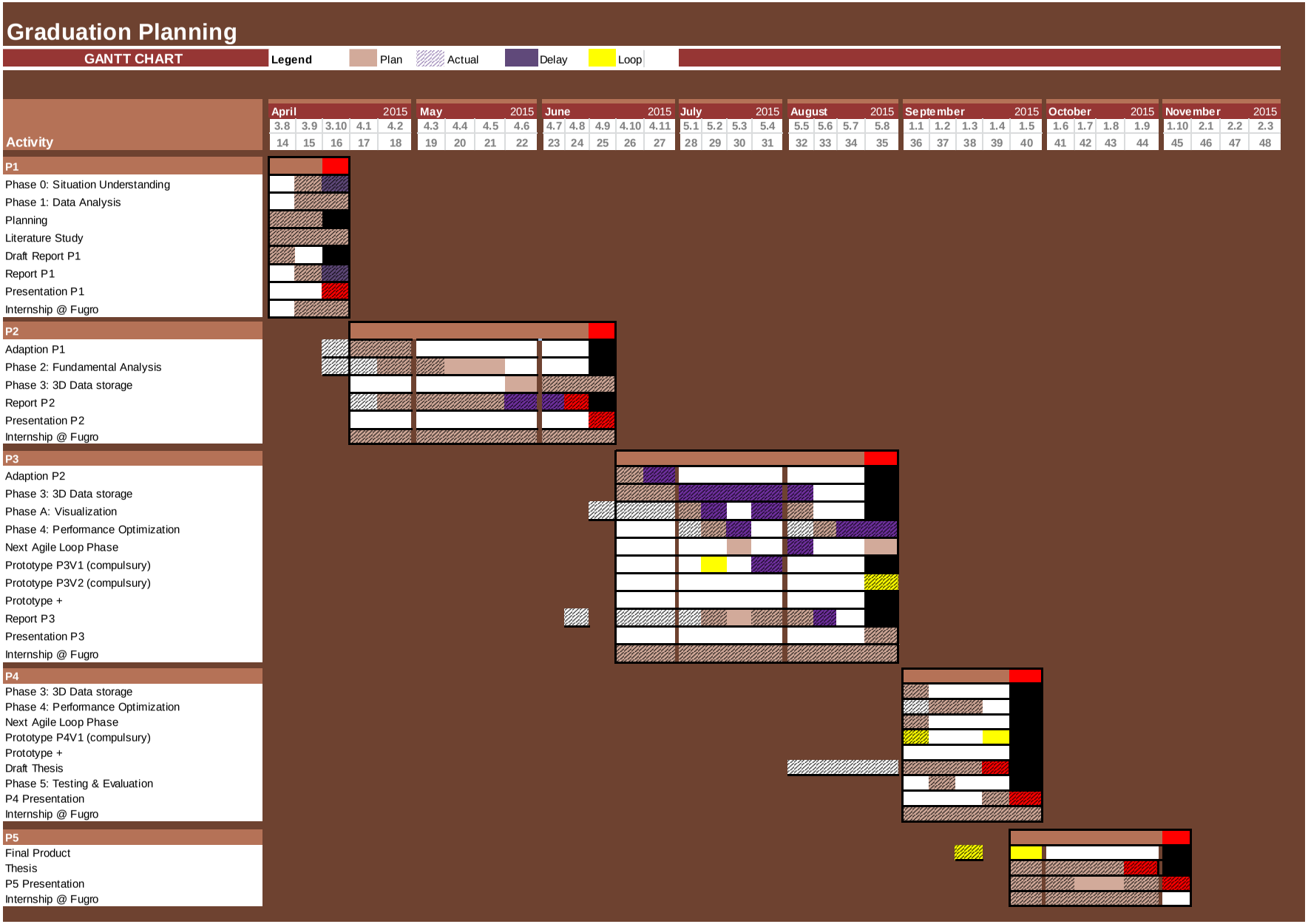


Figure C.1: Gantt planning of the graduation project.

During this graduation project the following equipment was used:

- Company Desktop
  - Dell Precision T1700
  - Windows 7
  - Intel Xeon 3.3 GHz
- Laptop
  - Acer V3-571G
  - Windows 8
  - Intel Core i7 3.2 GHz
- PostgreSQL 9.4.1
  - DBMS
  - Including the following extension(s):
    - PostGIS 2.2.0dev
    - SFCGAL 1.0.5
  - The DBMS for this graduation thesis used the test server of the TU Delft with a SFCGAL extension.
- Bentley MicroStation
  - CAD tool
- StarUML
  - UML tool
- Python
  - Scripting program
- Octaga
  - X3D visualizing program
- Modelnamings
  - Program developed by Fugro GeoServices which extracts parameters of CAD files into DBMS INSERT scripts.

E

Appendix: P5 Reflection

This thesis has proposed a methodology for storing solid CAD geometries in a 3D spatial DBMS. This 3D spatial DBMS can be a parametric DBMS or a non-parametric DBMS (converting the geometries to solid polyhedra). This graduation project has focused on reconstructing only one geometry type: cylinders. The experiment results show that a 3D non-parametric DBMS performs better, but needs more storage space compared to a 3D parametric DBMS. In order to minimize the storage space a 3D parametric DBMS is recommended, but some effort needs to be done.

The topic of this graduation project is related to the field of geomatics, mainly focusing on spatial DBMS management and Geo-Visualization. This graduation project has also incorporated knowledge regarding cartography (for creating illustrations of the benchmark process), sensing technologies (the process of capturing the building environment with a laser scanner), 3D positioning and location awareness (obtaining the location coordinated in a point cloud and performing CRS translations), 3D modelling (reconstructing the data set in 3D), quality of spatial analyses (computing the errors of the analyses) and Geo-Information Legislation (in order to maintain the privacy and copyright of the used data sets).

This graduation project is sponsored by Fugro GeoServices, which is also the provider of this graduation topic. Fugro GeoServices has provided technical support and data sets for conducting this research. By performing this graduation project for a company, more insights have been gained in the whole process of creating a 3D model of the petrochemical industry. As addition the general workflow in a GIS company has been experienced.

In the wider social context this graduation project has provided an ideal insight in changing the current workflow of companies. Many companies are only focusing on how to provide a solution to a customer with the existing resources and less on innovative solutions. This graduation project has provided future insights that the worlds of CAD and GIS can be interoperable by having a central DBMS. This central DBMS can be used by both worlds for their own purpose within their trusted software environment. However more research needs to be done if and how a DBMS can take a central role in the current workflows of companies.

Comparing the methodical line of approach of the Master Geomatics with the approach of this graduation project, both approaches start at conducting experiments on a smaller scale. When the experiments on the smaller scale are sufficient, the experiments are conducted on a larger scale. The smaller scale has equally been grown to a larger scale in order to notice trends for making predictions. Former studies are the key for obtaining knowledge if one does not have much knowledge about a subject.

Looking at the planning of this graduation project most of the delay has happened during the P3 period. This delay is caused by underestimating the length of some processes. In terms of planning this delay has a negative value, but in terms of knowledge gaining this is where most of the knowledge is gained. By focusing on ones errors, knowledge is gained in finding the cause of this error. In addition skills are improved by fixing these errors.

Comparing the amount of knowledge at the beginning of this graduation project with the knowledge at the end of this graduation project a lot of knowledge has been gained. This graduation project has given the opportunity to learn a new programming language (plSQL), to improve the knowledge in 3D spatial DBMSs and to use mathematical expressions and formulas (which was barely incorporated during the Bachelor of Architecture). Insights have been gained in how to develop a solution for a problem. Trial and error is the path one has to dare to come to a solution and to obtain knowledge.

References

3D-max. (n.d.). Point Surface. Retrieved 22th April, 2015, from http://www.3dmax-tutorials.com/Point_Surface.html

Adams, D. (2014). *Oracle Database Online Documentation 12c Release 1 (12.1). Application Development*

Arens, C. (2003). *Maintaining reality: modelling 3D spatial objects in a Geo-DBMS using a 3D primitive.* (MSc. Thesis), TU Delft, Delft University of Technology.

Ashdown, L., & Kyte, T. (2014). *Oracle Database Online Documentation 11g Release 2 (11.2): Database Administration* Retrieved from http://docs.oracle.com/cd/E11882_01/server.112/e40540/cncptdev.htm#CNCPT7654

Avin. (n.d.). Wilkommen in der Soup von Avin Fathulla. Retrieved April 22th, 2015, from http://avin.soup.io/since/298207348?mode=own

Ben-Kiki, O., Evans, C., & Ingerson, B. (2009). YAML Ain't Markup Language (YAML™) Version 1.1. *Working Draft 2008-05, 11*.

Benner, J., Geiger, A., & Leinemann, K. (2005). Flexible generation of semantic 3D building modeld. *Proceedings of the 1st Intern. Workshop in Next Generation 3D City Models*, 17-22.

Bhanu, B., & Ho, C.-C. (1987). CAD-based 3D object representation for robot vision. *Computer;(United States), 20*(8).

Blatov, V. A. (2006). Multipurpose crystallochemical analysis with the program package TOPOS. *IUCr CompComm Newsletter, 7*(4).

Boncz, P. A., Manegold, S., & Rittinger, J. (2005). *Updating the Pre/Post Plane in MonetDB/XQuery*: CWI. Information Systems [INS].

Bray, T., Paoli, J., Sperberg-McQueen, C. M., Maler, E., & Yergeau, F. (1998). Extensible markup language (XML). *World Wide Web Consortium Recommendation REC-xml-19980210. http://www. w3. org/TR/1998/REC-xml-19980210, 16*.

Brugman, B. (2010). *3D topological structure management within a DBMS, validating a topological volume.* (MSc. Thesis), TU Delft, Delft University of Technology.

Brutzman, D., & Daly, L. (2007). *X3D extensible 3D graphics for Web authors*. Amsterdam: Elsevier.

BuildingSMART. (n.d.-a). IFC4 DTV Objectives. Retrieved November 15th, 2015, from http://www.buildingsmart-tech.org/specifications/ifc-view-definition/ifc4-design-transfer-view/ifc4-dtv-objectives

BuildingSMART. (n.d.-b). IFC Geometric Model Resource. Retrieved April 29th, 2015, from http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcgeometricmodelresource/ifcgeometricmodelresource.htm

BuildingSMART. (n.d.-c). IfcSphere. Retrieved April 30th, 2015, from http://iaiweb.lbl.gov/Resources/IFC_Releases/R2x3_final/ifcgeometricmodelresource/lexical/ifcsphere.htm

Chorafas, D. N. (1983). *DBMS for distributed computers and networks*. New York: Petrocelli.

Clementini, E., Di Felice, P., & Van Oosterom, P. (1993). *A small set of formal topological relationships suitable for end-user interaction.* Paper presented at the Advances in Spatial Databases.

Computer Aided Detector Design. (n.d.). Boundary representation. Retrieved April 22th, 2015, from http://cadd.web.cern.ch/cadd/cad_geant_int/thesis/node23.html

Cowen, D. J. (1988). GIS versus CAD versus DBMS: what are the differences? *Photogrammatric Engineering and Remote Sensing, 54*(11), 1551-1555.

Crockford, D. (2013). The json data interchange format: Technical report, ECMA International, October.

Cromwell, P. R. (1997). *Polyhedra; 'one of the most charming chapters of geometry'*. Cambridge, UK: Cambridge University Press.

Ctech. (n.d.). Triangular Networks. Retrieved April 22th, 2015, from http://evshelp.ctech.com/Content/workbooks/fundamentals_of_visualizing_environmental_data/triangular_networks.htm
Date, C. J. (1975). *An introduction to database systems*. Reading: Addison-Wesley.
De Laat, R., & Van Berlo, L. (2011). Integration of BIM and GIS: The development of the CityGML GeoBIM extension *Advances in 3D geo-information sciences* (pp. 211-225): Springer.
Dollne, J., & Hagedorn, B. (2008). Integrating urban GIS, CAD, and BIM data by service-based virtual 3D city models. In M. Rumor, E. Fendel & S. Zlatanova (Eds.), *Urban and Regional Data Management*. London, UK: Taylor & Francis Group.
Donkers, S. (2013). *Automatic generation of CityGML LoD3 building models from IFC models.* (MSc. Thesis), TU Delft, Delft University of Technology.
Du, Q., Emelianenko, M., & Ju, L. (2006). Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations. *SIAM journal on numerical analysis, 44*(1), 102-119.
Du, Y., & Zlatanova, S. (2006). An approach for 3D visualization of pipelines *Innovations in 3D geo information systems* (pp. 501-517): Springer.
Eckel, G. (n.d.). *OpenGL Volumizer Programmer's Guide* C. Cary (Ed.) *Document Number: 007-3720-002*
Edoceo. (n.d.). Comma Separated Values (CSV) Standard File Format. Retrieved October 12th, 2015, from http://edoceo.com/utilitas/csv-file-format
Egenhofer, M. J. (1995). Topological relations in 3D: Technical report, University of Maine, Orono, USA.
Fagin, R., Kolaitis, P. G., Miller, R. J., & Popa, L. (2005). Data exchange: semantics and query answering. *Theoretical Computer Science, 336*(1), 89-124.
Fane, B. (2013). *AutoCAD 2014 for dummies*. Hoboken, NJ: John Wiley & Sons.
File-Extensions.org. (n.d.). COE file extension - Xilinx BRAM initialization file. Retrieved October 11th, 2015, from http://www.file-extensions.org/coe-file-extension
Finkelstein, E. (2014). *AutoCAD 2015 and AutoCAD LT 2015 Bible* (pp. 1299).
Fotheringham, S., & Rogerson, P. (2013). *Spatial analysis and GIS*: CRC Press.
Gellish. (n.d.). System Requirements for using Gellish. Retrieved May 18th, 2015, from http://www.gellish.net/index.php/knowledge-base/12-knowledge-base/16-faq-2.html
Goncalves, R., & Kersten, M. (2011). The data cyclotoon query processing scheme. *ACM Transactions n Database Systems (TODS), 36*(4).
Grasshopper. (2010). Grasshopper discussion forum: weaire phelan using galapagos. Retrieved May 20th, 2015, from http://www.grasshopper3d.com/forum/topics/weaire-phelan-using-galapagos
Groover, M., & Zimmers, E. (1983). *CAD/CAM: computer-aided design and manufacturing*: Pearson Education.
Guerrero Iñiguez, J. I. (2012). *Three-dimensional reconstruction of underground utilities for real-time visualization.* (MSc. Thesis), TU Delft, Delft University of Technology.
Hart, G. (1997). Calculating Canonical Polyhedra. Retrieved 22th April, 2015, from http://www.georgehart.com/canonical/canonical-supplement.html
Hassan, M., Ahmad-Nasruddin, M., Yaakop, I., & Abdul-Rahman, A. (2008). An integrated 3D cadastre–Malaysia as an example. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 37*(B4), 121-126.
Henrichs, M. (2009). *A Conceptual Framework for Constructing Distributed Object Libraries using Gellish.* TU Delft, Delft University of Technology.
Hernandez, C. R. B. (2006). Thinking parametric design: introducing parametric Gaudi. *Design Studies, 27*(3), 309-324.
Hijazi, I. (2011). *Integrated management of indoor and outdoor utilities by utilizing BIM and 3DGIS.* (Doctoral dissertion), University of Osnabrueck.
Hoff III, K. E., Keyser, J., Lin, M., Manocha, D., & Culver, T. (1999). *Fast computation of generalized Voronoi diagrams using graphics hardware.* Paper presented at the Proceedings of the 26th annual conference on Computer graphics and interactive techniques.

I-Chen, W., & Shang-Hsien, H. (2007). Trandormation from IFC data model fot GML data model: Methodology and tool development. *Journal of the Chinese Institute of Engineers, 30*(6), 1085-1090.
Intellidimension. (n.d.). Semantics.Server 2.0. Retrieved May 18th, 2015, from http://www.intellidimension.com/products/semantics-server/
Karimi, H. A., & Akinci, B. (2009). *CAD and GIS integration*: CRC Press.
Kaufman, A. (1989). *3D voxel-based graphics*. New York: Pergamon.
Kazar, B. M., Kothuri, R., van Oosterom, P., & Ravada, S. (2008). On valid and invalid three-dimensional geometries *Advances in 3D geoinformation systems* (pp. 19-46): Springer.
Kersting, O., & Döllner, J. (2002). *Interactive 3D visualization of vector data in GIS.* Paper presented at the Proceedings of the 10th ACM international symposium on Advances in geographic information systems.
Khuan, C. T., Abdul-Rahman, A., & Zlatanova, S. (2008). 3D solids and their management in DBMS *Advances in 3D geoinformation systems* (pp. 279-311): Springer.
Klyne, G., & Carroll, J. J. (2006). Resource description framework (RDF): Concepts and abstract syntax.
Kolbe, T., & Plumer, L. (2004). Bridging the Gap between GIS and CAAD Geometry, Referencing, Representations, Standards and Semantic Modelling. *GIM international, 18*, 12-38.
Kruip, C. J. H., Paardekooper, J. P., Clauwens, B. J. F., & Icke, V. (2010). Mathematical properties of the SimpleX algorithm. *Astronomy & Astrophysics, 515*, A78.
Kuipers, J. B. (1999). *Quaternions and Rotation Sequences. A primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton, New Jersey: Princeton University Press.
Laine, S. (2013). *A topological approach to voxelization.* Paper presented at the Computer Graphics Forum.
Ledoux, H. (2013). On the validation of solids represented with the international standards for geographic information. *Computer-Aided Civil and Infrastructure Engineering, 28*(9), 693-706.
Lee, J.-Y., & Koh, J.-H. (2007). A conceptual Data Model for a 3D Cadastre in Korea. *Journal of the Korean Society of Surveying, Geodesy, Photogrammetry and Cartography, 25*(6_1), 565-574.
Leslie, M. (2009). *Introduction to PostGIS* Retrieved from http://revenant.ca/www/postgis/workshop/index.html
Mederos, B., Velho, L., & De Figueiredo, L. (n.d.). *Moving Least Squares Multiresolution Surface Approximation*. IMPA–Instituto de Matematica Pura e Aplicada. Unpublished.
Microsoft. (n.d.-a). BULK INSERT (Transac SQL). Retrieved May 26th, 2015, from https://msdn.microsoft.com/en-us/library/ms188365.aspx
Microsoft. (n.d.-b). Import Data from a Data Feed. Retrieved May 18th, 2015, from https://msdn.microsoft.com/en-us/library/gg413490(v=sql.110).aspx
Microsoft. (n.d.-c). Relational Database Components. Retrieved April 24th, 2015, from https://technet.microsoft.com/en-us/library/aa174501(v=sql.80).aspx
Microsoft. (n.d.-d). Spatial Data (SQL Server). Retrieved April 24th, 2015, from https://msdn.microsoft.com/en-us/library/bb933790.aspx
Microsoft. (n.d.-e). XML Data (SQL Server). Retrieved May 18th, 2015, from https://msdn.microsoft.com/en-us/bb522446.aspx
Minh Duc, P. (2013). Self-organizing Structured RDF in MonetDB.
Mitra, N. J., Nguyen, A., & Guibas, L. (2004). Estimating surface normals in noisy point cloud data. *International Journal of Computational Geometry & Applications, 14*(04n05), 261-276.
Monedero, J. (2000). Parametric design: a review and some experiences. *Automation in Construction, 9*(4), 369-377.
MonetDB. (n.d.-a). Column store feature. Retrieved April 24th, 2015, from https://www.monetdb.org/Home/Features
MonetDB. (n.d.-b). Extending MonetDB with support for 3D Point Cloud and Voxels. Retrieved April 29th, 2015, from https://www.monetdbsolutions.com/extending-monetdb-support-3d-point-cloud-and-voxels

MonetDB. (n.d.-c). GeoSpatial. Retrieved April 24th, 2015, from https://www.monetdb.org/Documentation/Extensions/GIS
MonetDB. (n.d.-d). *MonetDB SQL Reference Manual*
Mossman. (2013). Videogames based in voxels. Retrieved April 22th, 2015, from http://mossman.es/videogames-based-in-voxels/
Movafagh, S. (1995). GIS/CAD Convergence Enhances Mapping Applications. *GIS World, 8*(5), 44-47.
MySQL. (2015). *MySQL 5.7 Reference Manual*
Nagel, C., Stadler, A., & Kolbe, T. (2009). Conceptual requirements for the automatic reconstruction of building information models from uninterpreted 3D models. *Proceedings of the International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 46-53.
Newell, R. G., & Sancha, T. L. (1990). The difference between CAD and GIS. *Computer-Aided Design, 22*(3), 131-135.
Object-e. (2009). Voronoi 3D v0.1.ms. Retrieved May 13th, 2015, from http://object-e.net/tools/voronoi-3d-v0-1-ms
OGC. (1999). Consortium Inc: OpenGIS Simple Feature Specification For SQL Version 1.1. *Open GIS project document*, 99-049.
OGC. (n.d.). CityGML. Retrieved October 12th, 2015, from http://www.opengeospatial.org/standards/citygml
Peng, J., & Kuo, C.-C. J. (2005). *Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition.* Paper presented at the ACM Transactions on Graphics (TOG).
Penninga, F., & Van Oosterom, P. (2008). A simplicial complex-based DBMS approach to 3D topographic data modelling. *International Journal of Geographical Information Science, 22*(7), 751-779.
Pharr, M., & Fernando, R. (2005). *Gpu gems 2: programming techniques for high-performance graphics and general-purpose computation*: Addison-Wesley Professional.
PostGIS. (n.d.-a). Box3D always gets cast to 2D geometry. Retrieved September, 8th, 2015, from https://trac.osgeo.org/postgis/ticket/103
PostGIS. (n.d.-b). Change BOX3D to be SRID and dimension aware. Retrieved November 14th, 2015, from https://trac.osgeo.org/postgis/ticket/1329
PostgreSQL. (n.d.). *PostgreSQL 9.4.1 Documentation*
Pu, S. (2005). *Managing freeform curves and surfaces in a spatial DBMS.* (MSc. Thesis), TU Delft, Delft University of Technology.
Qian, Z., & Woodbury, R. F. (2004). Between Reading & Authoring: Patterns of Digital Interpretation. *International Journal of Design Computing, 200*.
Ramakrishnnan, R. (2003). *Database management systems* (3rd ed.). Boston, USA: McGraw Hill Publisher.
Randal, P. (2008). SQL Server 2008: Spatial Indexes. Retrieved September 9th, 2015, from http://www.sqlskills.com/blogs/paul/sql-server-2008-spatial-indexes/
Rogers, D. (2001). *An introduction to NURBS with historical perspective*. San Francisco: Morgan Kaufmann.
Rossignac, J. (2002). *CSG-BRep duality and compression.* Paper presented at the Proceedings of the seventh ACM symposium on Solid modeling and applications.
SFCGAL. (2013). SFCGAL::Solid Class Reference. Retrieved April 29th, 2015, from http://oslandia.github.io/SFCGAL/doxygen/class_s_f_c_g_a_l_1_1_solid.html
Stippel, H. (1993). *Simulation der ionen-implantation*: na.
Stoter, J., & Salzmann, M. (2003). Towards a 3D cadastre: where do cadastral needs and technical possibilities meet? *Computers, environment and urban systems, 27*(4), 395-410.
Strobl, C. (2008). *Dimensionally Extended Nine-Intersection Model (DE-9IM)*: Springer US.
Stroud, I. (2006). *Boundary Representation Modelling Techniques*. London: Springer.
Teklemburg, J., Timmermans, H., & Borges, A. (1997). Design tools in a integrated CAD-GIS environment: space syntax as an example. *Decision support systems in urban planning. London: E & FN Spon*, 261-276.

The PostGIS Development Group. (n.d.). *PostGIS 2.1.8dev Manual SVN Revision (13470)* Retrieved from http://postgis.net/docs/manual-2.1/index.html
Van Lanen, H. A. J., Bregt, A. K., Randen, Y. v., & Hoosbeek, M. R. (1989). *Use of GIS, EC soil map, and land evaluation procedures to explore crop growth potential*. Wageningen: Winand Staring Centre.
Van Oosterom, P. (1985). *Visualiseren van aerodynamische gegevens (in Dutch).* (MSc. Thesis), TU Delft, Delft University of Technology.
Van Oosterom, P., Martinez-Rubi, O., Ivanova, M., Horhammer, M., Geringer, D., Ravada, S., . . . Gonçalves, R. (2015). Massive point cloud data management: design, implementation and execution of a point cloud benchmark. *Computers & Graphics*.
Van Oosterom, P., Stoter, J., & Jansen, E. (2005). 1 Bridging the Worlds of CAD and GIS. *Large-scale 3D data integration: challenges and opportunities*, 9-36.
Van Oosterom, P., Vertegaal, W., Van Hekken, M., & Vijlbrief, T. (1994). *Integrated 3D Modelling within a GIS.* Paper presented at the AGDM.
Vijlbrief, T., & van Oosterom, P. (1992). *The GEO++ system: An extensible GIS.* Paper presented at the Proc. 5th Intl. Symposium on Spatial Data Handling, Charleston, South Carolina.
W3C. (2003). RDF SQL HOWTO. Retrieved May 18th, 2015, from http://www.w3.org/1999/02/26-modules/User/RdfSQL-HOWTO.html
Walles, E. (2011). Csg.js. Retrieved 10th April, 2015, from https://github.com/evanw/csg.js/
Wang, Y., & Wang, S. (2010). *Research and implementation on spatial data storage and operation based on Hadoop platform.* Paper presented at the Geoscience and Remote Sensing (IITA-GRS), 2010 Second IITA International Conference on.
Wesselingh, S. (2007). *Visualization of a TEN (Tetrahedral Irregular Network) in a web client.* (MSc. Thesis), TU Delft, Delft University of Technology.
Xu, D. (2011). *Design and Implementation of Constraints for 3D Spatial Database: Using Climate City Campus Database as an Example* (MSc. Thesis), TU Delft, Delft University of Technology.
Zlatanova, S. (2000). *3D GIS for urban development.* (PhD. Thesis), ITC, The Netherlands.
Zlatanova, S. (2006). 3D geometries in spatial DBMS *Innovations in 3D geo information systems* (pp. 1-14): Springer.