Implementation of the Core Cadastral Domain Model in a Distributed Environment using OpenGIS Standards

Déo R. Rutamu
March, 2006
Implementation of the Core Cadastral Domain Model in a Distributed Environment using OpenGIS Standards

by

Déo R. Rutamu

Thesis submitted to the International Institute for Geo-information Science and Earth Observation in partial fulfilment of the requirements for the degree of Master of Science in Geo-informatics.

Thesis Assessment Board

Prof. Dr. Menno-Jan Kraak (Chairman), from GIP department ITC, Enschede
Ms. Drs. Marian de Vries (External examiner/supervisor), from GIS Technology of TU Delft
Dr. Andreas Wytzisk (First Supervisor), from GIP department ITC, Enschede
Ir. Christiaan Lemmen (Second Supervisor), from GIM department ITC, Enschede

INTERNATIONAL INSTITUTE FOR GEO-INFORMATION SCIENCE AND EARTH OBSERVATION
ENSCHÉDE, THE NETHERLANDS
Disclaimer

This document describes work undertaken as part of a programme of study at the International Institute for Geo-information Science and Earth Observation. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the institute.
Dedication

To Mom, Dad my first teachers, who inspired me to learn about the world and teach others: to my many teachers, professors, and advisors since, who helped mold me into what I am today and who has given me a new level of inspiration, support, and motivation. To all members of my family, having been in touch, made me to know where I do belong. May God grant me the privilege of making as positive and impact on others as you have all had on me.
Abstract

The proliferation of spatial data on the Internet is beginning to allow a much wider access to data currently available in various Geographic Information Systems (GIS). However, a major problem today is that important data is scattered throughout dozens of separately evolved distributed heterogeneous data sources, in a form that makes the “big picture” become difficult to obtain. Data integration presents a unified virtual view of all data within a domain, allowing the user to pose queries across the completed integrated schema without knowing the schemas of the underlying data sources.

GIS web services are services using the internet, for example to retrieve maps or query geospatial databases. In this context, the Open Geospatial Consortium (OGC) recently has developed several web specifications that should support interoperability between services and client of different vendors. One of these is the Web feature services specification. This specification is unique open standard in the sense that is the only specification that provides a standardized interface not only for editing geographic data (Transactional WFS) but also for querying (Basic WFS) services.

Since Web Feature services can be used to query data over the internet. Web Feature services should be able to process distributed query services by other Web services. Beside, the OGC efforts to lead interoperable standards, there are many other efforts from the international organization like (ISO, W3 and FIG), to provide common standards.

This thesis is an effort to a novel approach to design an architecture for data integration and distributed query. In this context, open web service architecture is conceived. Firstly, the integration of data sources are made by schema translation at the database level, using create view approach. Data sources views are created, using schema matching of the source into the global schema (core cadastral domain model) and secondly the resulting mediated schemas are wrapper in the Web feature services from the databases.

The propose architecture design has been carried out by a use-case driven approach considering cadastral data integration. Accordingly, requirements were derived from the use case and implemented in the underlying architecture design. Distributed queries like “Join” on two tables were successful done, although, most of the process were done manual, however distributed data sets have been successfully integrated. Finally we show how to make of the current open sources packages for the implementation of such architecture in a prototype for testing the distributed query.

**Key words:** Distributed environment, Distributed query, OpenGIS standards, Data integration, Core Cadastral Domain Model, Web Feature services.
Acknowledgements

It is amazing to look back and see how things have changed over the past few months, how others have influenced me. I have been blessed with a brilliant and incredibly generous group of collaborators and advisors, who given me great ideas and shaped my research ideas and my thesis work. Special thanks to my advisors, Dr. Andreas Wytzisk as my direct first supervisor, who let me be creative but keep me on track, and whom I have had countless stimulating discussions that led to new ideas. The constructive suggestions and discussions that I have got from Ir. C. Lemmen, my second supervisor, have improved my understanding to conceptualize the thesis, particularly on the cadastral issue and standardization. A special thank-you is warranted for Drs. Marian de Vries, who has been a good advisor and source of encouragement since the beginning – As my external supervisor, she helped solving technical problems, reviewing new pieces I wrote for this thesis and developing parts of the software. Many thanks for all my supervisors, especially for their expertise and pleasant way of working, created the right atmosphere to perform the research in.

The comments and supports during the thesis period from Prof. Dr. Menno-Jan Kraak, chairman of degree assessment board, and Dr. Corné van Elzakker, my advisor during the proposal-writing period, have motivated me to be more seriously in working on my project. I have come to understand just how critical this is in the research world. I would like to thank ITC giving me this opportunity to increase my knowledge. My sincere gratitude goes to GFM Program Director, G.C. Huurneman, and student advisor, Dr. V.A. Tolpekin for their encouragement and guidance. I also thank Dr. Tom Loran, all teachers and staff of ITC for their diligence and co-operation that have been helpful during the period of my stay at ITC.

I would like to extend my gratitude to the Netherlands Government for the Netherlands Fellowship Program (NFP), who dedicatedly agreed to shoulder all the financial responsibilities required for the M.Sc. study. Additionally, I will never forget, the encouragement coming from my institution and colleagues, from the Center CGIS-NUR, National University of Rwanda, Butare - Rwanda. As institutional bodies and people behind them, I would like to express my sincere gratitude for their support, especially Dr. Michele Schilling the director of the Center CGIS-NUR. I would also thank to all my classmates and all Rwandan friends; Didace Kayiranga, Ghislain Tenge, Rosena Charles Kibasa, and Coco Rulinda, whose company was enjoyable and memorable.
Table of contents

1. INTRODUCTION ................................................................................................................................. 1

1.1. Problem and research context ........................................................................................................ 1

1.1.1. Heterogeneity .............................................................................................................................. 1

1.1.2. Interoperability ............................................................................................................................ 2

1.1.3. The standardised cadastral domain model ................................................................................... 2

1.1.4. Interoperable standards ................................................................................................................. 3

1.2. Data Integration ........................................................................................................................................ 3

1.2.1. Data Warehousing ....................................................................................................................... 4

1.2.2. Query processing ......................................................................................................................... 6

1.3. Related Work to Data Integration ...................................................................................................... 6

1.3.1. Database Community ................................................................................................................... 7

1.3.2. Data integration system research projects .................................................................................... 7

1.4. Problem statement .................................................................................................................................... 9

1.5. Research identification ........................................................................................................................... 10

1.5.1. Research objectives ................................................................................................................... 10

1.5.2. Research questions ..................................................................................................................... 10

1.6. Project set-up .......................................................................................................................................... 11

1.6.1. Use-cases ................................................................................................................................... 11

1.6.2. Method adopted ........................................................................................................................... 12

1.6.3. Research requirements ............................................................................................................... 13

1.7. Thesis outline ........................................................................................................................................... 13

2. DISTRIBUTED COMPUTING ENVIRONMENT ...................................................................................... 17

2.1. Introduction .......................................................................................................................................... 17

2.2. Basics on Distributed Computing Systems ........................................................................................... 17

2.2.1. Distributed systems .................................................................................................................... 17

2.2.2. Client-server Architecture ......................................................................................................... 19

2.2.3. Parallel versus Distributed Technology ..................................................................................... 20

2.3. Distributed Web Services Architectures ............................................................................................... 22

2.3.1. Service Oriented Architecture (SOA) ........................................................................................ 22

2.3.2. Web Services Architecture (WSA) ............................................................................................ 25

2.3.3. OpenGIS Services Architecture (OSA) ..................................................................................... 26

2.4. Concluding remarks ............................................................................................................................... 29

3. INTEROPERABLE STANDARDS ............................................................................................................... 31
List of figures

Figure 1-1: Data integration architectures (warehousing and Integration compared) ........................................ 5
Figure 1-2: Two bottommost layers of the GiMoDig service architecture .......................................................... 8
Figure 1-3: SQL statement use-case query scenario example 1 ........................................................................ 12
Figure 1-4: SQL statement use-case query scenario for data integration .......................................................... 12
Figure 2-1: Example of Client-server Architectures ......................................................................................... 19
Figure 2-2: Some different database system architectures .................................................................................... 21
Figure 2-3: Element of Service-Oriented Architecture ......................................................................................... 23
Figure 2-4: Collaboration in SOA (Web services roles, operations and artifacts) .................................................. 24
Figure 2-5: Web Service Architecture Stack ....................................................................................................... 25
Figure 2-6: Viewpoints defined in ISO RM-ODP ................................................................................................. 28
Figure 2-7: OGC Web Services Architecture ....................................................................................................... 29
Figure 3-1: The OWS Service Framework (OSF) ................................................................................................. 36
Figure 3-2: Publish – Find – Bind pattern from ORM ............................................................................................. 37
Figure 3-3: Two examples of OGC filter encodings .............................................................................................. 40
Figure 3-4: Example WMS-request for retrieval of map image .............................................................................. 42
Figure 3-5: Web Feature Service ....................................................................................................................... 43
Figure 4-1: Framework for data integration: the main steps .................................................................................. 46
Figure 4-2: Binary and nary methods example ................................................................................................... 47
Figure 4-3: Schema integration strategies ........................................................................................................... 51
Figure 4-4: Classification of schema matching approaches .................................................................................... 55
Figure 5-1: 2 versions of expressing the UML relationship between person and land ........................................... 61
Figure 5-2: The administrative part of the CCDM ............................................................................................... 62
Figure 5-3: The legal part of the CCDM .............................................................................................................. 63
Figure 5-4: CCDM parcel registry ...................................................................................................................... 64
Figure 5-5: UML state diagram that illustrates how PartitionParcel is split ....................................................... 65
Figure 5-6: CCDM geography/topology part ....................................................................................................... 66
Figure 5-7: Attributes assigned to the three main tables of the CCDM .............................................................. 68
Figure 5-8: A portion of the UML model Dutch cadastral database-basic classes ............................................... 68
Figure 6-1: Wrapper/mediator architecture ......................................................................................................... 76
Figure 6-2: Syntactic conflict examples .............................................................................................................. 77
Figure 6-3: Semantic conflict example. The attribute Phone is optional in site 1 & mandatory in site 2 .............. 78
Figure 6-4: Instance conflict example ................................................................................................................ 79
Figure 6-5: Schema integration as a set of schema transformation sequences (T1 and T2) ................................. 80
Figure 6-6: Mapping the Dutch cadastral model into the Core domain model (CCDM) ....................................... 81
Figure 6-7: Create view statement for data integration ......................................................................................... 82
Figure 6-8: Generic data integration architecture ................................................................................................. 84
Figure 6-9: Detailed data Integration Architecture ............................................................................................... 85
Figure 6-10: Web feature Services interoperability for data integration ............................................................ 86
Figure 7-1: in uDig, documentation states that " ................................................................................................. 91
Figure 7-2: The developed WFS client interface ................................................................................................. 92
Figure 7-3: Create view general syntax definition ............................................................................................... 93
Figure 7-4: Schema translation using the “create view” database approach .......................................................... 94
Figure 7-5: Approach to data integration at the database layer ........................................................................... 95
List of tables

Table 3-1: Comparison of the most important operations of WMS, WCS and WFS ............................................. 44
Table 5-1: the following considerations are based on the Person table association ............................................ 69
Table 5-2: the following considerations are based on the Right association table............................................. 69
Table 5-3: the following considerations are based on the Parcel table association............................................. 70
1. Introduction

1.1. Problem and research context

Today, one of the new emerging technologies in Geo-ICT is the federation concept for improving access to the distributed databases in land administration and spatial data infrastructure [1]. In developing countries, many cadastral data namely land ownership, cadastre, topography and property tax are organized and distributed under the different agencies including tax information. Access to these data sets is very difficult because they are in the analogue forms. Since maintenance of these data sets is time-consuming and tedious in analogue forms, many of them normally reliability and quality of these data is hence questionable to the user community.

Many countries are now starting to convert the cadastral data into the structured digital databases for quality products and services. Geo-ICT including Web technology now allows us efficient and easy access to these data at the place where they are required. In this way, location does not matter where the data are stored [2]. However, from the management perspectives, the data are better kept at the source where the data is gathered as maintained. Integrated cadastral data is required by the large variety of users such as municipalities, real estate brokers, provincial government, insurance companies, financial agencies (including bank), utility companies who provides public services, ministries, courts, individual citizens, notaries, etc.

In addition, organizations maintain and provide specialized data according to their functions; however, in many cases users or even organizations themselves need other datasets for particular application [3]. Data integration between different systems is not straightforward and time consuming [4]. Likewise, proprietary data formats are a major obstruction for data integration. There is no accepted standard for spatial or geographic data representation. Each GIS provides its own proprietary format as well as its specific query language. Indeed, geographic resources like cadastral data are deigned for a variety of different purposes [5]. Orthogonal directions in the design of geographic resources may affect the semantics of the data they contain and impair their integration. These discrepancies make the integration of geographic resources significantly complex.

1.1.1. Heterogeneity

Heterogeneity among databases is caused by the design autonomy of their owners in developing such systems. Each system was typically designed to support local requirements, under constraints imposed with a given system. We can distinguish several types of heterogeneity [6] the platform, Database Management System (DMS), location and semantics level. The platform level copes with the fact that databases reside on different brands of hardware, under different operating systems, and interacting through various network protocols. Levelling these differences leads to platform independence. DMS level independence allows programmers to ignore the technical detail of data implementation in a definite family of models or among different data models.
Representing data with different data models creates heterogeneity because of the inheriting expressive powers and limitations of DMS data models [7]. Location independence isolates the user from knowing where the data reside. Current technologies such as de facto standards (e.g., ODBC\(^1\) and JDBC\(^2\)), now ensure a high level of platform independence at a reasonable cost, so that this level can be ignored from now on. DMS level independence is effective for some families of DBMS (e.g. through ODBC or JDBC for RDB\(^3\)), but the general problem is still unsolved when several DMS models are to cooperate. Location independence is addressed either by specific DBMS (e.g. distributed RDBMS) or through distributed object managers such as CORBA\(^4\) middleware products. Finally, semantic level independence solves the problem of multiple, replicated and conflicting representations of similar facts. Despite much effort spent by the scientific community, semantic independence still is an open and largely unsolved problem [8-10].

1.1.2. Interoperability

To address the problem of interoperability of information systems in general, the term mediation has been defined [11] as a service that links data resources and application programs. Tasks involved in mediation include [12] (1) accessing and retrieving relevant data from multiple heterogeneous sources; (2) transforming retrieved data to be integrated; (3) integrated the homogenized data; (4) managing the instance and structural conflicts; and (5) reducing the integrated data by abstraction. Several prototype mediator systems have been developed (e.g., [12], [13]). These approaches usually rely on wrappers to access data sources and to retrieve and translate the results into some common integrated representation. In addition, data warehouses often use wrappers to import data from remote sources that are then materialized locally, and queries are evaluate locally against the warehoused data. The key disadvantage of the warehouse approach is the need for local administrators to maintain the data while it provides control over the contents of the warehoused data. Mediators, on the other hand, submit the queries directly to the wrappers, and integrate the results locally. Accessed data is up-to date, but data access can be costly. Warehousing approach to data integration will be discussed in the next section 1.3.

1.1.3. The standardised cadastral domain model

To integrate data means to provide one global view over different data sources. This view can be either materialized or virtual. An important thing is to combine data in meaningful way and let them be accessible as one whole. There are two main problems resulting from the data integration. The first is data modelling (how to integrate different sources schemas); the second is their querying (how to answer to the queries posed on the global schema). The solution of these partial problems depends on many conditions, among them is if the global view is materialized or virtual and how much information should be supported in the global source.

To facilitate the exchange of cadastral information between states or provinces, a standard model for the cadastral domain is proposed by [14]. The standardized Core Cadastral Domain Model (CCDM) reflects aspects found in most cadastral systems and describes these according to international

\(^{1}\) Open Database Connectivity
\(^{2}\) Java Database Connectivity
\(^{3}\) Relational Database System
\(^{4}\) Common Object Request Broker Architecture
standards. The purpose of the cadastral domain model is to represent general modelling knowledge related to cadastral issues that can be extended to specific requirements, but two different areas portray the main advantages of this CCDM. On one hand, it represents the core components of cadastral systems, and on the other hand, it supports the exchange of the data between heterogeneous cadastre systems of different countries or regions within one country [15].

In understanding data sharing concept, the integration of cadastral data from various distributed databases is a priority so that the users get consistent result with a least effort on data handling. Again, the development of the CCDM is a valuable contribution for standardisation in the cadastral domain. Additional use of the CCDM, apart from standardising the cadastral domain, is the example it sets for general development of cadastral systems. Countries with its cadastral system in transition can here look at the model as a core to found their own cadastral model on [15].

1.1.4. Interoperable standards

Today, however, most spatial data handling applications on the Internet concern Web mapping or Web cartography offering functionality for the use, distribution and production of maps by means of the Internet [16, 17]. Additionally, they allow for visualizing spatial data and submitting simple queries. Current standardization efforts such as the initiative by the Open GIS Consortium (OGC) support this type of geospatial data handling. OGC released the Web Map Service Implementation Specification (WMS) which standardizes the way map images, service-level metadata, and information about particular map features contained in a map are requested [18]. Nevertheless, an OGC-compliant Web map server does not necessarily include any further tools for spatial analysis and modelling. As data are returned in an image format, they cannot be accessed for additional processing.

The need for exchanging and sharing spatial data on the Internet that goes beyond the transfer of query results as Web maps is well recognized. International Standardizations (e.g. OGC and ISO) and the World Wide Web Consortium (W3C) stress more on the exchange of geographic data in a format that enables further client-side processing. Additionally, software producers such as ESRI with the recently released ArcGIS Server are extending their Web map server products by processing functions that consist of advanced spatial data handling operations. Besides commercial developments, research efforts concerning Web-based GIS applications also aim at offering advanced spatial data handling capabilities.

Focusing on GIS tools Anderson and Moreno-Sanchez [19] demonstrates the implementation of spatial analysis capabilities around open specifications and open source software. Results show that both open specifications and open source software libraries have become powerful and mature enough to be applied in Web-GIS projects. Maximal interoperability is achieved by strictly conforming to the guidelines of open specifications [20].

1.2. Data Integration

Over the past years, the basic operating paradigm for data processing has evolved as computing technology itself has changed [21]. We have moved from mainframe-based, centralized data management systems to networks of powerful PC clients, group servers and the internet. Recent trends in research suggest that we may ultimately be moving to an even more extreme, peer-based
model in which all machines both consume and provide data and computation in a full decentralized architecture [22].

Motivation for these changes has come not merely from more advanced hardware and networking technologies, but from a natural desire to decentralized system generally form a bottleneck in term of scaling performance, but the centralized computing model can be a scalability bottleneck in terms of administration[23]. As stressed in the previous section, when data is “owned” and managed by numerous heterogeneous groups with different needs, a central schema is difficult to design, it typical relies on the development of standards before it can be constructed, and it is slow to adapt to the need of its members. A decentralized collection of autonomous systems, however, can be much more dynamic, as individual components can be separately designed and redesigned to meet the needs of their small user populations[24].

It is unsurprising, then, that today most enterprises, institutions, and formal collaborations – which typically are comprised of groups that are at least partly autonomous from one another, seldom operate with only centralized, common data management systems [1]. Instead, individual group often create their own separate systems and databases, each with the schema and data most relevant to their own needs. Moreover, an organization’s databases seldom represent all of the data it owns or accesses: in many cases, additional data is encoded in other formats such as documents, spreadsheets, or custom applications, and often today’s organizations have collaborations with external entities (or make acquisitions of new groups) that may share certain data items.

Unfortunately, the common data management model, a decentralized collection of autonomous, heterogeneous systems typically suffers from a major shortcoming. There is no longer a single point of access to organizational data that can be queried and analyzed in a comprehensive form. The decentralized computation model provides great flexibility, but the centralized model provides uniformity and a global perspective[25].

Two solutions have been proposed to this problem, both of which are end-points along a broad continuum of possible implementations: data warehousing lies at one end of the spectrum, and “virtual” data integration at the other[26]. Both approaches take a set of pre-existing decentralized data sources related to a particular domain, they develop a single unified (mediated) schema for that domain, and transformations or source mappings are specified to describe the relationship between each data source and the mediated schema [27]. Then a series of transformation or source mapping are specified to describe the relationship between each data source and the mediated schema.

1.2.1. Data Warehousing

In data warehousing, the expectation is that the data changes infrequently or that the integrated view does not need to be current, and that large numbers of expensive queries will be posed over the integrated view of the data. Hence, the full contents of the global schema are precomputed (by evaluating all source mappings), they are stored in a separate “warehouse” database that will used for querying, and significant attention is given to physical warehouse design indexing, in order to get the best possible and done offline, using ETL (Extract, Transform, and Load) tools (see, Figure 1-1 (a)).

Data integration addressed the case where warehousing is impractical, overly expensive, or impossible: for instance, when queries only access small portions of the data, the data changes
frequently, “live” data is required, data-providing partners are only willing to grant access to their data, or the global schema itself may be changed frequently.

In virtual data integration, the global schema is strictly a logical or virtual entity, queries posed over it are dynamically rewritten at runtime to refer to actual data sources (based on the source mapping), and data is fetched from the source (via wrappers) and combined. Data integration has become of increasing interest in recent years as it has matured, because it has several benefits to the implementer versus warehousing [28]: it support data sources that may only allow limited access to data; it supports a “live” view of the data environment.

One potential drawback of the virtual data integration approach is that certain data cleaning and semantic matching operations between sources are too expensive to perform “on-the-fly”, and must be computed offline; another is that virtual data integration may heavily load the data sources. To handle
these issues, an implementation whose characteristics fall between the full-precomputed model of the data warehouse and the full virtual model of data integration may be desirable: certain data items or matching information may be cached, prefetched, or precomputed. In this thesis, we will only consider, the full virtual data integration approach to fall within the general category of data integration, but our interests lie in handling distributed query through the virtual mediated schema (see, Figure 1-1 (b)).

The Figure 1-1 (a) shows the data warehousing, replicates data from sources offline and executes its queries over the warehoused data and on Virtual data integration (b) presents a virtual, mediated schema but fetches the actual data on-demand from the underlying data sources.

1.2.2. Query processing

Until very recently, the emphasis of research in data integration was on developing models [29], [30], mappings [31], [32], and translators or “wrappers” for data sources [33]. With, additional work on the problem of translating or “reformulating” ([34], [35, 36]; [37], [38]) queries over the mediated schema into queries over the real sources. These problems have been addressed well enough to provide a level of functionality that is sufficient for solving many real-world problems. Now that there are established algorithms and methodologies for data integration, there are two important challenges remaining to be addressed to make data integration a widespread technology. Firstly, is the problem of defining correspondences between entities at different data sources (i.e., mappings between different concepts or schema items, but also mapping entities that appear in different sources with different representations); secondly is the problem of developing system-building techniques that allow a data integration system to perform well enough that it can be useful in practice. Other research have started to address aspects of the first problem [32], [39], [40]; my focus is on the combination of both problems, emphasising on cadastral heterogeneous database integration and distributed query execution.

As discussed earlier, traditional query processing deals with an environment on which statistics are computed offline and used to optimize a query, which can then be executed. This generally is effective in a standard database environment because the data and computing environment are under the strict control of the database management system (DBMS), and they are thus fairly predictable and consistent. Yet even in this context, many simplifying assumption must be made during optimisation, and there are many situation in which the traditional model does poorly (e.g., in many circumstances, the optimizer accumulates substantial error in modelling complex queries).

Even worse, the data integration domain has a number of features that make it substantially harder than the conventional DBMS context. Data integration typically interacts with autonomous sources and externally controlled networks: it becomes difficult to model the expected performance of such data sources (sometimes even the amount of data it will return is unknown), and the performance of the network may be unexpectedly bad. Unpredictability and inconsistency are the norm in querying remote, autonomous data sources.

1.3. Related Work to Data Integration

Data integration has been a topic of study that actually dates back to the early 1980s [41]. Autonomy and schema heterogeneity are often characteristics associated with federated databases, or the
federated capabilities now built into Microsoft SQL server). However, key differentiating characteristics of data integration systems include the ability to work with non-database data source (Web sources) as well as data sources not built to work together: data integration is typically retrofitted over existing sources that were never designed to interchange data with other sources in different schemas.

Additionally, [42] gives a useful overview of the research questions in data integration that are still not solved. With the Internet, there is now an extra dimension: the technology to access, retrieve and query information on remote servers exists [43]. However, because of the loose coupling of all this information, the fact that the user group is not known and the unpredictable nature of the queries that will be posed, the information integration issue has only become more imminent. Indeed, Many research disciplines are involved in data integration: from computer science and database research to artificial intelligence, the Semantic Web Description Logic [43, 44].

### 1.3.1. Database Community

The database community had been working on efforts to distributed and federated databases for many years even before the Web phenomenon. Early efforts included distributed databases such as Distributed INGRES [45], which extended the basic techniques of INGRES for databases distributed across LANs. Mariposa was a descendent of INGRES that attempted to distribute to wide-area, and it made use of an economic model of bidding for services as its basic infrastructure; unfortunately, the model was not tremendously successful[46]. Instead, the database industry developed simpler distributed versions of Oracle, DB2, and Informix. Additionally, standards such as ODBC and OLE-DB [46] were proposed for allowing for limited interoperability between databases and certain types of data exchange with applications.

### 1.3.2. Data integration system research projects

First data integration systems, appeared in the very early 90s, but as with the Artificial Intelligence community, it was the advent of the Web that triggered significant interest in integrating data[47]. Significant projects included the Stanford TSIMMIS [48] project, which was one of the first systems to propose a semi-structured data model as a means of schema mediation; Hermes [49], IRODB [50], and DISCO [51, 52], which generally focused on learning expected performance levels from sources and on handling sources failures; and Carnot and Infos-leuth [53], which were data integration systems for deductive databases. All of these efforts used the global-as-view approach to schema mediation: the mediated schema is defined as a view over the data sources. This has the benefit of allowing simple “unfolding” of the views (view definitions can be macro-expanded into the query), but it has a trade-off that the mediated schema may have to be revised anytime a new source is added, and it has difficulty representing incomplete data sources.

The Information Manifold [34] was significant because it popularized the local-as-view (LaV) formalism, which gave significantly greater flexibility (addition of data sources generally does not require revisions to the mediated schema, and incomplete data sources are easy to model) at the cost of new query reformulation step. The authors of the system proposed the bucket algorithm as a way of

---

5 Note that a distributed Database is neither heterogeneous nor autonomous: the Database admin partitions data across a set of servers that all run the same database software and contain portions of the same Database.
reformulating mediated queries to refer to underlying sources. (Note that Razor, which appeared slightly later, was the corresponding system from the AI community that made use of local-as-view. It used the inverse rules algorithm or [54] to (reformulate queries). The Tukwila system [55] uses the same basic approach as these systems, but focuses on the problem of providing their capabilities at high performance and at scale.

Accordingly, the European Union have initiates the GiMoDig\(^6\) project [56], [57] which serves as a real-time data integration and generalization for delivering geospatial data to a mobile user. The project aims at the creation of a seamless data service providing access, through a common interface, to the primary topographic geo-databases maintained by the National Mapping Agencies (NMAs) in various countries [58]. Therefore, our approach to data integration and distributed query form different heterogeneous sources is embedded in the overall context of the GiMoDig data integration service layer. In addition, in the GiMoDig project more emphasis are put on providing appropriately generalized map data to the user depending on a mobile terminal with limited display capabilities (i.e., Handle devices such PDA’s, Smart Cell phones). For the purpose of this research, the next section provides the GiMoDig system architecture.

The GiModig architecture is based on a layered service stack, in which a service would make queries to the service below it, do some processing on the data received as a response, and provide the results of this processing as a service to the layer above it [59]. On the first, level the data providers (i.e. in the GiMoDig, the NMAs) run a Data Service providing raw data in an XML-encoded form (Cfr, Figure 1-2). Above the data service is the Data Integration Service layer. This layer works as a bridge between the local data models and the GiMoDig global schema on one hand and between the national coordinate systems and the common European reference system on the other.

\[\text{Figure 1-2: Two bottommost layers of the GiMoDig service architecture}\quad [60]\]

\(^6\) Geospatial info-mobility service by real-time data-integration and generalisation (GiMoDig).
Similarly, for the purpose of this research, we also aim at improving the exchange of cadastral information between municipal or provincial distributed national cadastral databases systems using a common data model in order to integrate and query the data through the common interface (global schema), but there are many differences as well as similarities from the technical point of view.

Additionally, [43] proposed a query translator approach for cadastral data integration across-border. Their approach uses concepts and techniques from the Semantic Web, in the form of using an ontology language to specify the correspondences and relation between the data models of the cadastral systems of different countries and the core model. However, they do not use a semantic reasoner that computes semantic relations ‘on-the-fly’. Indeed, the semantic relations between the models are established beforehand, during the conformance verification process [43].

Furthermore, on the third level of the GiMoDig, architecture is the Data Processing layer. This layer is responsible for various data processing and analysis tasks, like map generalisation or dynamic map labelling [56]. The fourth layer in the system architecture is the Portal Service. The main tasks of this layer can be listed as: process the service requests coming from the client, subsequently forwarding the request in an appropriate form to the Data Processing layer below, and transform the resulting piece of geospatial data into a visual representation (visual map image, i.e. SVG). The fifth layer is the so-called Value-Added Service Layer. It controls the creation of the map to be delivered to client applications, taking into account the parameters related to the adaptive map display.

On the sixth layer are finally the client applications. An advantage of the layered architecture approach is that the results can be adapted to a wide set of different client environments (e.g. the traditional Web browsing on a portable PC platform, the more restricted Web accessed on PDA devices and the client application on mobile cell phones).

In contrast to this approach to integration via an intermediary model or format, there is currently also much research into another strategy for data integration, such as peer-to-peer data integration (see, [42], [22, 61]. Peer-to-peer data integration approach is shortly covered in the last section on chapter 4.

1.4. Problem statement

Geographic data is increasingly becoming available on the Internet, allowing a large audience to access and share the rich databases that are currently used by GIS people such as municipalities that would like to exchange cadastral data. This situation is an opportunity for the emergence of a GIS Web community. However, GIS data is dramatically heterogeneous, being available in various formats, annotated using a variety of methods, and stored in disparate media (flat files, relational or object-oriented database, etc.). There is no accepted standard for spatial or geographic data representation/exchange. A typical example could be for instance, municipalities within even a country may have developed their own cadastral systems according to a specific purpose.

In addition, as a combination, different designs, approaches, and several data model have been designed for different purposes, and implemented using heterogeneous databases systems and hence integration and querying execution problems have emerged. Orthogonal directions in the design of geographic resources may affect the semantics of the data and impair their integration. These discrepancies make the integration of different geographic resources significantly complex. Furthermore, taking into account the scale of data integration, the often complex and heterogeneous
distributed query processing and domain specific computational capabilities supported by these sources makes GIS integration a challenge.

Do the current interoperable standards from the international organisations efforts (ISO, OGC, FIG) mature enough to facilitate cross system transactions, and integration for distributed queries? Nevertheless, with increased heterogeneity the standardisation of data exchange, especially spatial data, becomes important. However, commercial spatial applications have been slow and sometimes reluctant to integrate many of the spatial standards and specifications proposed by international organizations (like ISO and OpenGIS), emphasising on own solutions and innovations.

Taking into consideration:

- the current data exchange interoperable standards offered by the emerged and promising OpenGIS community efforts and recognizing the importance of data integration as being ubiquitous in most of cadastral projects and,

- the need for exchanging cadastral information across border and between neighbour municipality, also the need for acquiring a nationwide cadastral system according to the core cadastral domain model as developed by the initiative of “International Federation of Surveyors” – FIG, the following question arises:

*How to apply and extend available interoperable standards and specifications to support the integration of distributed heterogeneous cadastral databases, particularly how to execute query in a distributed environment?*

### 1.5. Research identification

#### 1.5.1. Research objectives

Based on the above raised question, the following objectives may be formulated as the central intention of the research:

- Identify the proper interoperable standards, methods, and technologies, capable to solve the issue of merging disparate data sources, which are based on the “FIG” core cadastral domain model.
- Design an architecture for the integration of distributed cadastral databases.
- Implement a limited prototype and architecture using open-source software tools, emphasising on integration and query execution on distributed heterogeneous cadastral databases.

#### 1.5.2. Research questions

Based on the proposal for solution, the following main research question was formed as the central intention of the research:

*How to apply and extend available interoperable standards and specifications to support the integration of distributed heterogeneous cadastral databases, particularly how to execute query in a distributed environment?*

This main question can furthermore be subdivided into several sub-questions:

1. What are the suitable interoperable standards and the feasible methods for integrating distributed heterogeneous data sets? What have to be extended?
2. What are the use-case requirements to be fulfilled for such data integration architecture?

3. How can be used the core cadastral domain model as basis for data exchange and integration, in order to facilitate query execution on heterogeneous distributed environment?

4. What is a suitable approach to data integration and distributed query to fulfill the identified requirements?

1.6. Project set-up

1.6.1. Use-cases

As stated before, the need for dynamically combining data from several sources for analyzing the data has grown. Integrating data in this context not only applies to retrieving the data and combining it, but it can also include improving data, and GIS availability, as well as simplifying data exploration and analysis. The following use case scenario, but realistic examples illustrate two scenarios that emphasize on typical cadastral data integration issues.

Two municipalities want to integrate cadastral data in order to check for some cadastre administrative issues related to some particular parcel owners. The database integration example comprises two independent heterogeneous databases both describing aspects of a municipal cadastre registry information system that are required to interoperate.

Consider for example a data integration system that helps users find parcels on the Dutch cadastre system databases. The system provides the uniform interface in terms of mediated schema, which is a virtual schema that captures the relevant aspects of the cadastral domain. The mediated schema may contain elements such as Person, Right, and Parcel, listing person right and parcel respectively. The system maintains for each data source a source schema that describes the content of the data source.

Scenario 1: Given a user query formulated against the mediated schema, such as “i.e., find parcels that belong to person name= ‘Paula’”, the system translates the query into query in the source schema and executes these queries with the help of the wrappers, the combines the data returned by the sources to produce the final answers. To translate user queries, a data integration system uses semantic mapping between the mediated schema and the local schemas of the data sources. Today, the system builder specifies such semantic mappings manually.

To illustrate this example, let firstly, consider that the parcel geometry and administrative data are kept in two different databases. A typical distributed query scenario would be to get the parcel data from one Dataset through WFS and the administrative data from another through the other WFS. This would mean in relational database terms – a ‘join’ between two tables.

Scenario 2: Secondly, considering a more general scenario, based on the main example would be the integration of two data sets distributed from two municipal cadastral databases, which have two different data models. Hence, query the data (using the two WFS separately) using the global search criteria and display the output of the first wrapper (WFS1) and the output of the second wrapper (WFS2) in one ‘list’. In this case (mapping schemas to the CCDM), you have to make a ‘union’ in database terms.
Examples of queries (e.g., Figure 1-3 and Figure 1-4)

Core Model:

```sql
SELECT name, address, type_of_right FROM naturalperson, right
WHERE person.id = right.person_id AND municipality = '…'
```

**Figure 1-3:** SQL statement use-case query scenario example 1

Here, the complicating factor is not the different names for classes and/or attributes, but different (names for) associations between classes plus knowledge about the join attributed (foreign keys) that must be used. However, in the second case (Core model and the Dutch model) it might no be trivial to rewrite the query based on the semantic relations formalized in the mediated schema using database views. One of the research questions is therefore whether the creating databases views is a useful approach to data integration and to distributed query such as joins between tables.

Dutch model:

```sql
SELECT p.gesl_naam AS family_name, p.naam_niet_nat AS company_name
FROM xfio_parcel AS x, object_parcel AS o, mo_objectadres AS m,
mo_recht AS r, mo_subject AS p
WHERE x.x_aksr_objectnummer = '…'
AND x.x_aksr_objectnummer = o.g_aksr_objectnummer
AND o.x_aksr_objectnummer = m.x_aksr_objectnummer
AND m.x_aksr_objectnummer = r.x_aksr_objectnummer
AND r.gerechtigde = p.subject_id;
```

**Figure 1-4:** SQL statement use-case query scenario for data integration

### 1.6.2. Method adopted

The research was initially preceded through the literature review and the analytical work to identify proper available interoperable standards for data exchange and data integration. The study forwards with the in-depth literature study of distributed system environments (architectures and their functionalities) and an overview study on data integration approaches. Indeed, the identification of these interoperable standards, and their improvement for the development of new standards limitations as well as the application of schema-mapping techniques using views approach for data integration as core of the integration approach.

The study identified the prerequisites for the design of a system cadastral data integration of architecture through literature review, use-case model, and basic requirements. In that context, an Open interoperable Web architecture has conceived. Additionally, based on the use case requirements the underlying architecture functionalities of the prototype have been proposed, in term of standards, data exchange format and for the designed solution for distributed data integration and exchange
through the Web. For example, it had adopted the OGC standard framework for distributed interoperable Geo-services, interfaces, (e.g. WFS interface specifications), and W3C Web Services framework standard for specifying the system components and the prototype architecture implementation.

Finally, as a proof of concept, the study try to validate the result by a use-case based example – execute query in a distributed environment, and implement the generic architecture framework for data integration using OpenGIS standards and Open source licensed packages.

1.6.3. Research requirements

1.6.3.1. Data

This research would experiment with two cadastral datasets from the Netherlands cadastre, and test the usability of proposed interoperable Geo-service architecture for cadastral data integration and distributed query execution based on the core cadastral domain model as foundation for cadastral databases integration.

The selected datasets from the Netherlands cadastre are two datasets containing one table with the parcel geometry and a number of tables with the administrative data. The geometry type is polygon. Actually, this is the only difference with the actual Dutch cadastral database now: the real Dutch cadastral database does not contain polygons but linestrings (the parcel boundaries) and points (for the placement of the text labels with the parcel numbers). The location is part of the municipality of Gouda. Apart from these two data sets, this research would have to use also, the Dutch data model and the CCDM model from the FIG in order to define the mediated schema or global schema, which helps in querying the different sources using views on the databases through the wrappers (WFS).

1.6.3.2. Software

The tools used to carry out this research are manifold, with the objective to use mainly free or open-source licensed software whenever possible. Cadastral datasets from the Netherlands Cadastre are available, but need to be converted from Oracle database format into PostGIS spatial database format. Therefore, we have been using:

- SQL Manager 2005 Lite for PostgreSQL editing,
- uDig v. 10.5, for editing and visualization of data,
- PostgreSQL 8.1, which contain PostGIS, as extension will be used as spatial database management system and finally,
- Geoserver 1.3.0 RC6, will serve as Web Feature Services. The testing on Geoserver has occurred on JDK 1.4.2.
- SVG plug-in

1.7. Thesis outline

This thesis consists of eight chapters, including this introductory chapter.
1.7.1.1. Chapter 1: Introduction

This chapter introduces the subject matter that we are going to discuss in depth. The main topics are background, problems, objectives, research questions, scope, and thesis structure.

1.7.1.2. Chapter 2: Overview of distributed computing environments

This chapter reviews the state-of-the-art distributed environments, including the basics of distributed geo databases systems, and geoprocessing interoperability issues. The Web Services, OpenGIS Web Services Architecture are covered in the study as well as distributed systems architecture in general.

1.7.1.3. Chapter 3: Interoperable Standards

Interoperable geoprocessing, and interoperability, and Open standards are key-issue in distributed computing and data integration in particular, for the purpose of this research a particular attention is been doing on interoperable standards (GML XML and Filter encoding) as well as OpenGIS web service interfaces and particularly web feature services.

1.7.1.4. Chapter 4: Data integration concepts

This chapter defines the fundamental concepts and approaches related to data integration issues. It first reviews and describe data integration problems and some of the proposed solutions. Additionally, different approaches to data integration are discussed. It focuses on the structural data integration approach including answering query using views for data integration (the local and global as view) approach are compared and analysed.

1.7.1.5. Chapter 5: Core Cadastral Domain Model

The core cadastral data model is proposed as an interoperable standard for supporting the integration of distributed cadastral data. Additionally, the data set used for the use-case are analysed and prepared for integration.

1.7.1.6. Chapter 6: Integration Architecture for Cadastral data

This chapter first, proposes an overview of the integration architecture. Accordingly, the preconditions and requirements that have to be fulfilled based on the use-case scenario for designing an architecture for cadastral data integration are identified. Secondly, the general principles behind the techniques of mapping heterogeneous schemas are being recall and syntheses from chapter 4. It provides a review of the components, standards and methods; composing the integration strategy, pointing out their principal role and their interrelation, keeping the overall scene of the integration framework in perspective. Furthermore, the processes of data conversion and data model translation are being analyzed. Finally, according to the identified requirements an architecture for distributed cadastral data integration is proposed.

1.7.1.7. Chapter 7: Proof of concepts (prototyping on use case)

This chapter start describing the proof of concepts for testing the data integration and distributed query according to the use case scenario. It discusses the system architecture and the general aspects of the prototype implementation for data integration. Results discussions of the main functionalities of
the WFS testing, in order to facilitate distributed query execution based on the use case scenario are discussed. Finally, The Open source packages and applications are discussed.

1.7.1.8. Chapter 8: Conclusions and recommendations

This chapter concludes the research. The conclusion comprises some general and specific contribution to the research. The future research part presents recommendations for further research on this area.
2. Distributed Computing Environment

2.1. Introduction

Out of the need to access and change information resources of other computer systems, distributed computing has arisen [62]. A distributed computing system consists of a number of processing elements, not necessarily homogeneous, that are interconnected by a computer network, and that cooperate in performing certain assigned tasks [23]. As a general goal, distributed computing systems partition a big, unmanageable problem into smaller pieces and solve it efficiently in a coordinated manner. The economic viability of this approach stems from two reasons [24]: (1) more computer power is harnessed to solve a complex task, and (2) each autonomous processing element can be managed independently and developed its own applications.

There are many challenges to overcome in successfully designing a distributed computing system. As stated above, the main goal of a distributed computing system is to connect users and resources in a transparent, open, and scalable way. Ideally, this arrangement is drastically more fault tolerant and more powerful than many combinations of stand-alone computer systems. Indeed, there are two approaches to solving the problems of the distribution and heterogeneity of spatial data (Donaubauer, 2004). For the purpose of the research the following approached are considered: (1) Data integration approach, which will be covered in chapter 4 and (2) Interoperability by means OGC Web Services (these systems just give answers to request that are issued via standardized web interfaces). The latter is partially covered in this chapter as well as in next chapter related to Interoperable standards.

This chapter deals with issues of distributed computing environments and particularly distributed System Architectures. The main subject covered in this chapter is particularly, distributed system architectures, after providing a short introduction. This chapter first defines distributed systems including the basics on distributed systems on section 2.2., including client-server architecture, and advantages of distributed databases. Distributed systems architecture including Service oriented architecture, Web services architecture, and OpenGIS Web Service architecture are provided in section 2.3. Finally, section 2.4 gives final remarks.

2.2. Basics on Distributed Computing Systems

2.2.1. Distributed systems

In general, a distributed system allows its components (e.g. users, pc’s, databases) to cooperate in some way. Sharing, managing data, applications and operations are examples of this. Several formal definitions of a distributed system exist.

Worboys [63] gives a short definition which describes what is meant with distributed systems in this research. For such a distributed system, the components can for instance share data or operations:
A distributed system is a collection of autonomous computers linked in a network together with software that will support integration.

For other definitions, especially on types of distributed databases, see [64, 65] and web distributed computing ([66], [67], [68]). An important characteristic of a distributed system is whether it is homogeneous or heterogeneous. Homogeneous refers to that each of the distributed databases employs the same DBMS software, data model and architecture so translation of data is unnecessary within the distributed system. Heterogeneous distributed systems contrary consist of several DBMS, each potentially employing own software, data model and architecture. Federating such a system is thus more complex as the data has to be translated to shared format before being served as a content of one integral database. Geography Markup Language (GML) as specified by the OGC is intended to serve this role for exchange of spatial data.

Two main types of heterogeneity are observed. DBMS heterogeneity that is due to differences of DBMS, like e.g. differences in data model, data structure, constraints and query language. Secondly is semantic heterogeneity, which is caused by semantic and ontological differences of DBMS modellers. This is e.g. differences in hierarchies, classes, geometry and attributes.

Data may be distributed among multiple databases in different ways [69]. Three types of distributed databases are discussed here in a little more detail: replicated databases, federated databases and independent data servers. If multiple copies of some or all the data are stored at multiple sites, the database is called a replicated database [70]. The main reason for using replicated data is to increase data availability. By storing critical data at multiple sites, the database can operate even though some sites have failed. It also improves performance of retrieval for global queries, because the result of such a query can be obtained locally since there are many copies of each data item, as compared to a single copy database [71]. This benefit is mitigated by the need to update all copies of each data item. In addition, keeping the copies consistent is also an issue. Thus, retrieving data is faster; editing data is slower [72], [69].

Another type of distributed database is the federated database system. Sheth and Larson [69] treat federated databases extensively. Accordingly, [73] give the following description: federated databases are separate databases that are structured, perhaps with middleware or special database access software, in such a way that they can be queried as a single database. Federated databases are used when already existing databases, spread amongst different organisations, must co-operate or interoperate [66]. Therefore, there must be agreement on the overall database schema and on which different locations manage which different parts of the data.

Influenced by the growth of the Internet, databases are more and more accessible over the Internet. These databases often are independent of each other, but still can serve a similar purpose or in the same domain [68]. In other words, these databases act as independent data servers that can be accessed by the end-user to combine data [67]. In such an infrastructure, end-users should be able to search for the appropriate data by searching the meta-data data providers publish for their databases in catalogues. Berg [74], describe such a model in the context of Internet GIS. An important advantage of this model is that data are managed at the source. There is no need anymore to manage copies of the data, which reduces the risk of inconsistencies in the database(s).
2.2.2. Client-server Architecture

In the context of architecture for distributed database management systems (Distributed DBMS), the client-server architecture is natural and widespread [68]. This hold also for internet-based systems - as a special case of distributed systems, (i.e. in a very large network). Accordingly, [25] argue for that, there exist three different types of distributed DBMS architectures. Namely: Client-Server System; Collaborating Server System; and Middleware System (Figure 2-1).

Client-server systems have gained much popularity with the introduction of the web and bettered communications. Its popularity is foremost gained by its explicitly, simplicity and usability. At the backend there is a server that connects to the databases and services their content to a network which number of clients is connected to. The clients (front end) can be diverse software that receives data from servers and delivers to users through customised interfaces. Therefore can data stored as raw text, number and binary codes in a database, be viewed by user as nicely drawn map with user friendly editor to manipulate the data.

![Figure 2-1: Example of Client-server Architectures](image-url)

(a) conventional relation between database and users, (b) the Client-Server System, (c) Collaborating Server System, and (d) Middleware System.

The main drawback of distributed client-server system is that it cannot easily fulfil the condition of distributed data independency as defined earlier. It is maybe possible to develop a client that translates spatial queries into sub-queries for relevant underlying databases and returns appropriate results. It is however difficult when submitting complex query string that overlaps the databases, as the query has to be run separately on each database. The clients also get complicated if they need to be capable of constructing sub-queries to be executed in different databases and then piecing them together again. In a way, they would then be starting to act as servers themselves.
This problem is approached with the collaborative server system architecture (Figure 2-1c). Here is the client level skipped and the user interacts directly through his interface to a local server, which furthermore is a component of larger network of servers all connected to their own local database. Each server can only commit transactions on its local database. For accessing external data it is capable to structure and direct sub-queries to the other databases, as a result of user query, and piece together again for the user to interpret.

As an example, we could imagine that there were four cadastral districts in a country each operating their own database, but served federated in accordance to collaborative server system architecture. Now a user on one office makes a query on its data that includes parcels stored in the other databases. The procedure could be as follows:

- User makes a query to its server (I): give me all parcels in the country larger than 4 ha
- Server I receives the query and by inspecting the query discovers its scope and as a result constructs a sub-query that is directed to the three other database servers;
- Each of the servers runs the corresponding query on their databases and returns the results to server
- Server I receives the results, pieces it together and returns to user, as it was its own.

Finally, to mention is the middleware system architecture that releases the servers of this burden of constructing sub-queries and directing them to other servers. A middleware mirrors the underlying servers along with performing a lot of work that otherwise would be committed at the server side. This could be for example locking mechanisms, data constraints and validation, logging etc. Figure 2-1d illustrates this, as users cannot connect to the databases without through the customised middleware that directs theirs queries and transactions to corresponding databases.

### 2.2.3. Parallel versus Distributed Technology

We can define a distributed database (DDB) as a collection of multiple logically interrelated databases distributed over a computer network, and a distributed database management system (DDBMS) as a software that manages a distributed database while making the distribution transparent to user.\(^7\)

Turning our attention to parallel system architecture, there two main types of multi-processor system architectures that are commonplace [23]:

- Shared memory (*tightly coupled*) architecture: Multiple processors share secondary (disk) storage and share primary memory.
- Shared disk (*loosely coupled*) architecture: Multiple processors shared secondary (disk) but each has their own primary memory.

These architectures enable processor to communicate without the overhead of exchanging messages over a network.\(^8\) Databases developed using the above types of architectures are termed *parallel database management systems* rather than DDBMS, since they utilize parallel processor technology. Another type of multiprocessor architecture is called *shared nothing architecture*. In this architecture,

---

\(^7\) This definition and some the discussion in this section are mainly based on [212].

\(^8\) If both primary and secondary memories are shared, the architecture is also know as shared *everything architecture*. 
every processor has its own primary and secondary (disk) memory, no common memory exists, and the processors communicate over a high-speed interconnection network (bus or switch). Although the shared nothing architecture resembles a distributed database computing environment, major differences exist in the mode of operation. In shared nothing multiprocessor systems, there is symmetry and homogeneity of nodes; this is not true of distributed database environment where heterogeneity of hardware and operating system at each node is very common. Shared nothing architecture is also considered as an environment for parallel databases. Figure 2-2 contrasts these different architectures.

**Figure 2-2:** Some different database system architectures.
Adapted from [23]

---

9 **Bus:** A network in which all nodes are connected to a single wire (the bus) that has two endpoints. Ethernet 10Base-2 and 10Base-5 networks, for example, are bus networks and **Switch:** are fundamental part of many networks. They speed things up. Switches allow different nodes (a network connection point, typically a computer) of a network to communicate directly with one another in a smooth and efficient manner.
(a) Shared nothing architecture. (b) A networked architecture with centralized database at one of the sites. (c) A truly distributed architecture.

2.3. Distributed Web Services Architectures

As stated in the previous section, emergence of more powerful computers and network infrastructure put forward the computing paradigm from tightly coupled monolithic centralized system to loosely coupled distributed computing system across networks. The distributed computing system works properly when the application components, which executes on different computers throughout a network are able to communicate [75]. In the 1990s, the big software vendors such as Object Management Group (OMG), Microsoft, Sun Microsystems, International Business Machine Corporation (IBM), etc., began to develop their technologies to enable communication among distributed components [76]. The technologies of OMG’s Common Object Request Broker Architecture (CORBA), Microsoft Distributed Component Object Model (DCOM), Sun Microsystems’s Remote Method Invocation (RMI), Distributed Smalltalk, and IBM’s Distributed Services Object Model (DSOM) are able to communicate programs running in different locations in the network environment [77].

The uses of these technologies are being limited to facilitate the business-process integration and automation due to tightly coupling of components. In order to face the challenges of interoperability, heterogeneity and ever-changing requirements, there is a need of application that has to provide the characteristics of loosely coupled, location transparent and protocol independent computing platform.

In order to get the concept of Web distributed geoprocessing system, this research focuses on the latest developments of software architectures built in the notion and concept of services. These service-oriented software architectures are the heart of distributed processing. The following sections provide the brief overview of Service-Oriented Architecture (SOA), OGC Services Architecture, and more particularly Web Feature Services (WFS).

2.3.1. Service Oriented Architecture (SOA)

2.3.1.1. Web Services and SOA

Web services are technologies that allow applications to communicate with each other in a platform- and programming language-independent manner. A Web service is a software interface that describes a collection of operations that can be accessed over the network through standardized XML messaging. It uses protocols based on the XML language to describe an operation to execute or data to exchange with another Web service [78]. A group of Web services interacting together in this manner defines a particular Web service application in a Service-Oriented Architecture (SOA).

While Web services allow dynamic features to combine multiple services into applications, you still have to build the services first. Programming languages in computer science are continually evolving. People, began decades ago with the idea of a function in which you provide some parameters, it executes some operation on those parameters, and it returns a value based on its calculations. Eventually, this first concept evolved into the object where each object had not just a number of functions it can perform, but also its own private data variables, rather than relying on external system-wide data variables that previously made it more complex to develop applications[79]. As applications began communicating with each other, the concept of defined universal interfaces for
objects became important, allowing objects on other platforms to communicate even if they were written in other programming languages and ran on other operating systems.

Accordingly, today’s IT challenges are to provide flexible and responsible platform for distributed computing to the ever-changing business environment, make those heterogeneous systems and applications communicate as seamlessly as possible within the platform. In this context, a special kind of software architecture [80] known as Service-Oriented Architecture (SOA) exhibits these characteristics disciplined use of service-orientation and provides a platform to accomplish those achievements.

A SOA is a form of distributed system architecture with typical properties such as message orientation, description orientation, network orientation, and platform independent [81]. According to [82], [83], and [84] the services in SOA provide the functionality of discoverable and dynamically bound, self-contained and modular, stress interoperability, loosely-couple, network-addressable interface, coarse-grained interfaces, location-transparent, composite modules, comprised of components, and support self-healing.

Figure 2-3: Element of Service-Oriented Architecture

[85]

The major levels of abstractions within SOA are operations, services and business processes[86]. These levels deliver the application functionality either in the form of services to end-use applications or other services through distributed systems. The architectural stacks and elements of SOA can be broadly classified into functional and non-functional characteristics (Figure 2-3). The functional aspect of SOA comprises from transport, protocol, description and registry components of the services (see left pane of Figure 2-3) whereas the non-functional or quality aspect of SOA consists policy, security, transaction, and management of the services (see right of the pane of Figure 2-3) [85]. The most important aspect of SOA is that it separates the implementation of the service from its interface [80].

The SOA follows the “publish, Find and Bind” operation paradigm known as service trading (see Figure 2-4), which addresses the discovery of available service instances, i.e., publication of service descriptions, lookup / finding of service descriptions and binding or invoking of services based on service descriptions [87]. The interaction of such trading can be decomposed into three categories of
components of fundamental roles: a service provider, and a consumer of service on a logical level [88] within a distributing computing network. The service consumer finds dynamic service location by querying the service registry for compatible services. The service provider makes the service registry for compatible services.

The service provider makes the service available with its service contract and also advertises a service matching with its criteria. If the service exists, the service consumer and the service provider can interact with each other. Therefore, the service modeling is related with identifying the right services, organizing them in a manageable hierarchy of composite services, choreographing them together to support a business process [89].

Service-Oriented is the fundamental evolution in software engineering. The service consumer further does not need to concern with the implementation of the service, because of the condition provided that the required functionality and quality of service meet the criteria. At the same time, the service provider offers a perspective on how to design the realization of the component that offers the services; its architectural decision and designs [89]. According to [90], [91], [92], the benefits of SOA can be summarized into the following points:

- Leverage initial investment,
- It is easier to integrate and manage complexity,
- More responsive and faster time-to-market,
- Reduces cost and increases re-use,
- Be ready for what lies ahead, and
- Risk mitigation, Technology and vendor independence, and
- Clarity of application topology.

Therefore, the SOA provides a suitable architecture of the modern enterprise of distributed computing paradigm.

![Service-Oriented Architecture](image)

**Figure 2-4: Collaboration in SOA (Web services roles, operations and artifacts)**
2.3.2. Web Services Architecture (WSA)

Web Services architectures (WSA) are the fundamental building blocks in the move to distributed computing on the Internet, which implements the SOA. There are many different kinds of published WSA that are realizations of SOA. Most of them are product architectures forwarded by major enterprise-software vendors. The generic Web Service architectures are still being developed by major standardization organizations [93]. Each vendor, standard organization, marketing research firm are trying to define Web Services in a slightly different way according to their business and Web Services strategies. Consequently, not only WSA stack varies from one organization to another but also the number and complexity of layers for the stack depends on the organization [94]. Therefore, it is important to differentiate vendor product architecture from generic Web Services architecture in term of framework and methodology.

A generic Web Services architecture regardless of vendor products, offers a repeatable and consistent way to design and deploy scalable, reliable Web Services [95]. One of the aims of this research is to provide the concept of distributed computing environment in form of Web Services Architecture, along with WSA stack. Therefore, the following sub-section provides the Web Services Architecture stack in a generic form, as well as the OpenGIS Web Service Architecture.

2.3.2.1. Web Services Architecture Stack

Web Services are accepted as a key implementation of SOA, because it offers a distributed computing approach for integrating extremely heterogeneous applications over the Internet. Similarly, Web Services architecture provides coherent view of Web Services components by incorporating many layered and interrelated technologies [81]. These technologies can be presented in many ways by forming a Web Services architecture stack. The conceptual levels of those stacks provided by, [96], [97], and [87]) are similar in many aspects.

![Web Service Architecture Stack](image)

**Figure 2-5: Web Service Architecture Stack**

[97]
Accordingly, Dubray [97] has been presented a generic conceptual Web Services architecture stack. Figure 2.5, shows taxonomy of a Web Services architecture stack. The specifications are classified into two axes. The horizontal axis represents the specification’s applicability during run-time, design and both, whereas the vertical axis demonstrates the three fundamental concepts of modern distributed computing paradigm such as Messages, Services-Oriented Architecture (SOA). It can be seen that the capabilities of upper layer are dependent on the lower layers. Likewise, the vertical towers represent the requirements that must be addressed to build the stack at every level [87].

The technologies which play a critical role in this architecture stack are XML, SOAP, WSDL and UDDI [67], [68], [98]. These core technologies are accepted and implemented quite uniformly as open standard. However, the higher-level component of the architecture that defines the strategic perspective of business processes remains an open problem [96]. The development of open standards of these unsolved technological problems will help to strengthen the Web Services architecture.

2.3.3. OpenGIS Services Architecture (OSA)

The Open Geospatial Consortium Inc. (OGC) and ISO Technical Committee 211 (ISO/TC 211) have jointly developed an international standard for geographic services architecture. The architecture provides a framework for developers to create software facilitating the users to access and process geospatial data across a generic computing interface from a variety of an Open information technology environment [99]. The OGC Abstract Specification, topic 12: OpenGIS Services Architecture is based on the ISO/DIS 19119 [99], geographic information services.

The OpenGIS Services Architecture adopts ISO Reference Model of Open Distributed Processing (ISO/IEC 10746) (RM-ODP) [100] that is defined in ISO 19101 [101] as an interoperability reference model. The ISO 19101 as a geographic information reference model defines an Extended Open Systems Environment (EOSE) model for geographic services. Therefore, OpenGIS Services Architecture also adopts the EOSE’s architecture reference model [68].

Similarly, the OGC Basic Services Model (BSM) as an implementation of the ISO/TC 211 services architecture as found in ISO 19119 Geographic Information - Services [101], has been evolved into a General Services Model (GSM) to specify the OWS common elements. In this context, the following sections provide the OpenGIS Services Architecture from the ISO Reference Model of Open Distributed Processing (ISO/IEC 10746) and the OGC Basic Services Model perspective respectively.

2.3.3.1. Reference Open Distributed Processing

Framework for the ongoing work of the Open Geospatial Consortium and the basis for recent and future specification work. ORM defines different Viewpoints to specify an information system respectively to derive relevant standards for setting up an IS (based on ISO Reference Model for Open Distributed Processing; RM-ODP). According to [100], the architecture is composed of a set of components, connections, and topologies that are defined through a series of viewpoints. According to ISO RM-ODP, the ISO/DIS 19119 provides five viewpoints (see Figure 2-6): Enterprise, Computational, Information, Engineering, and Technology viewpoints.
2.3.3.2. **The enterprise viewpoint**

The enterprise viewpoint articulates a business model that defines the purpose, scope, and policies of an enterprise or business organization. The enterprise viewpoint highlights the role of Geospatial location as a fundamental ingredient for modelling the world. It provides a representative value chain of Geospatial information to create a value-add Geospatial-based product within an enterprise and information community, and caters the major requirements on OGC technology that are derived form value chain [102].

2.3.3.3. **The computational viewpoint**

The computational viewpoint describes the functional decomposition of the system into a set of services that interact at interfaces and represent information sources and sinks. The computational viewpoint is considered as a basis for service chaining that defines the services with reusable interfaces, service-metadata, service/data coupling, and service chaining. The computational viewpoint of Open Reference Model (ORM) [102] not only defines the functional decomposition of the system into a set of services that interact at interfaces also describes service classification scheme and OWS Service Framework (OSF).

These are three types of service classification schemes: semantic classification – using service taxonomy, interface classification – using the set of operations at an interface, and capability classification – using service metadata. Likewise, the OSF is a profile of the OGC Services taxonomy that identifies services, interfaces, and exchange protocols that can be used by any application [102] to support publish, discovery, and binding of services through service metadata.

2.3.3.4. **Information viewpoint**

Information viewpoint describes information objects that are saved, processed, and referenced by information sources and sink. The information viewpoint is based on the EOSE model of the geographic information as defined by ISO 1901. Indeed, the EOSE architecture reference model provides geographic services taxonomy in six classes: human interaction services, model/information management services, workflow/ task services, processing services communication services and systems management services.

2.3.3.5. **The engineering viewpoint**

Likewise, the engineering viewpoint focuses on mechanisms for distributed transparencies (distribution structure), and support services across networks. Indeed, engineering viewpoint describes the interaction between computational objects in the sense of clustering computational functions at physical nodes of a communications network. The engineering viewpoint provides an OGC Web Services framework.
2.3.3.6. The technological viewpoint

The technological viewpoint is concerned with the choice of technology or infrastructure in the distributed system, which provides a basis for cross platform interoperability. This infrastructure ensures the objects to inter-operate within the framework of distributed computing platform. The approach of ISO 19119 to cross platform interoperability is through a development of platform-neutral, platform-specific service specifications, which are the basis for multiple-platform-specific specifications. Accordingly, OGC has defined a suite of Web Service interface for HTTP bindings [103], for invoking operations of a service: GET [104] and POST [105]. Likewise, WSDL was used in the first phase of OGC Web Services of XML Capabilities document in the Open Signature portion [102].

2.3.3.7. OGC Basic Services Model

The OGC Interoperability Program 2000 (IP200) – comprising Web Mapping Testbed phase 2 (WMT2) and the Geospatial Fusion Services (GFS) testbed, has further enlarge interoperability by enhancing the Web Map Server (WMS) specification and Service Catalogs. These services – collectively called OGC Web Services, differ in their purposes and details of specifications but share a number of common elements.

In this context, the BSM [104], describes the basic service framework of OWS and specifies the common elements. The OGC BSM – An OGC Discussion Paper is an implementation of the ISO/TC 211 Services architectures. This discussion paper presents the OGC Web Services architecture (see, the above Figure 2.6) at conceptual level. The OGC also recommends the 3-tier approach. The OGC Web Services Architecture (OWS) clarifies how these perspectives combine to form the specific components in an OGC context.
The OWS includes three principal types of geo-referenced information access services, (Web Map Server (WMS), Web Coverage Server (WCS) and Web Feature Server (WFS)). In addition, there are services such as GeoParser and GeoCoder that return spatially referenced results. Following is an architecture diagram showing conceptually how these services are related. In the Section 3.3 are provided more details on these OWS and their comparison.

2.4. Concluding remarks

This chapter has dealt with geospatial Distributed Environments and particularly distributed services architectures. The emerging technologies in distributed systems architectures and distributed services, suitable for delivering Geo-service provide promising new tools and technologies in distributed computing environment. In this context, the evolution of distributed computing and its architecture to meet the today’s challenges has been analyzed in particular Service-Oriented Architecture (SOA) and Web Services Architecture (WSA) approach. The WSA provides a means of interoperability within a distributed computing architecture.

Likewise, the architecture and its implementation form as a SOA provides the interoperability and extension to be implemented with other technologies like OpenGIS Services Architecture. One of these type of services, Web Feature Services, should make editing and update of geographic data in a distributed, heterogeneous environment possible over the Internet in order to integrate and share data in a multi-user environment, interoperable geoprocessing and GIS Web Services are important subjects in this context. Interoperable standards and OpenGIS Web Services Interfaces are subject of the next chapter.
3. Interoperable Standards

3.1. Introduction

To share data over networks, system components have to work together – interoperate - in some way. Open Systems use standards that support this interoperability. This topic is dealt in section 3.2. OpenGIS Web Services including OpenGIS Service Framework are covered in section 3.3. Relevant interoperable technologies and standards are discussed in section 3.4. Section 3.5, discusses OpenGIS Web services interfaces, including Web Map Services (WMS), Web Coverage Services (WCS) and Web Feature Services (WFS) and finally, section 3.6, concludes this chapter with final remarks.

3.2. Interoperability

From the previous chapter it may be clear that there is a need to share and integrate (geographical) data from multiple, heterogeneous database systems. In order to do so, the systems need to interoperate in some way. Uitermark [106] distinguishes two different levels of interoperability. There is a technical level or, the systems perspective, with an understanding of information processing issues, like network protocols and standards for data set files. And there is a semantics level or, the data modeling perspective with an understanding of the semantics of information processing. In this particular chapter the technical level is more considered. We define interoperability according to [73]:

> Interoperability is the ability for a system or components of a system to provide information portability and interapplication, cooperative process control. Interoperability, in the context of the OpenGIS Specification, is software components operating reciprocally (working with each other) to overcome tedious batch conversion tasks, import/export obstacles, and distributed resource access barriers imposed by heterogeneous processing environments and heterogeneous data.

Interoperability refers to the capability for applications running on dissimilar computers to exchange information and operate cooperatively using this information [107]. [107] also defines application interoperability. With application interoperability they mean that two, or more, spatial information systems can exchange data seamlessly and users can query remote databases without any knowledge of their underlying data models. This makes chaining of services from different sources possible. This concept is discussed in section 2.2.3.

Portability refers to the ability to move from one system to another, without having to make (major) changes. [73] define, for example, user portability as “the ability of a user to move from one system to another without having to learn everything again”. Simultaneously, information portability could be described as the ability of a system to exchange information or data with another system.

[107] relate portability to soft- and hardware only: according to them portability refers to the capability for software to run on different types of hardware. As said before, the ability to edit geographic data (i.e. besides retrieving, also creating, updating and deleting geographic features) is
essential. [73] definition of interoperable geoprocessing makes clear what interoperable systems in the context of GIS should be able to do:

“Interoperable geoprocessing” refers to the ability of digital systems to 1) freely exchange all kinds of spatial information about the Earth and about objects and phenomena on, above, and below the Earth’s surface; and 2) cooperatively, over networks, run software capable of manipulating such information.

Thus, interoperable geoprocessing is concerned with retrieving and editing geodata. Because of the complexity of current computing systems and because of systems being heterogeneous, standardization is required to realize interoperable systems [66], [108]. An important characteristic of a standard is that a standard provides - among other things – a break-down of the complexity, and more importantly, an agreed break-down. [109] argues that these standards are not just for external use. It can be necessary to provide "internal" standards when designing complex systems, creating syntactically and semantically well-defined interfaces between systems, problem domains, development groups and enterprises.

3.2.1. Problems with interoperability

The problems in sharing information in Geo-Information Infrastructures (GII) and interoperability can be subsumed under two main categories [107]:

- **Political, Institutional and Economic problems**: The provision of information for the public requires legal considerations (e.g., copyright), proper institutional organization and data access rules, as well as pricing schemes.

- **Technical problems**: These include devising techniques for the provision of up-to-date inventory of the available data, mechanisms for seamlessly sharing information, update and consistency constraints and semantics.

The technical problems are the most relevant for this thesis. [107] divide the technical problems in two categories:

- Data modeling (which includes data migration and integration), and
- System architecture

According to [108], the heterogeneities between the component datasets in a distributed system lead to problems that are common both in data migration and integration. These problems need to be tackled, to make the component systems interoperable. The heterogeneities that cause the problems exist on various levels, such as:

- Different computer hardware and operating systems.
- Different DBMS software.
- Different concepts for data modeling (e.g. relational models, object-oriented models).
- Semantic heterogeneities.

The first three types are syntactical in nature. They occur at the software and hardware level, while the semantic heterogeneities usually occur at the application level [107]. For problems concerning
data modeling, this subdivision in syntactical and semantic problems is common (see, [107], and [108]). However, differences exist in classifying problems as syntactical or semantical. Syntax concerns the format (the language, so to speak) in which information is represented. In the context of geographic information Bishr [107] discusses syntactic constructs. Syntactic constructs include spatial primitives to represent the geometry (e.g., nodes, arcs and surfaces) and topological relationships (e.g., interior, exterior or neighborhood) of real world features. It also includes constructs to represent the thematic properties and relationships of the real world features (e.g., classes, subclasses, aggregation, associations, generalization or attributes).

Syntactic problems arise when differences exist in encoding the data [108], i.e. when the syntactic constructs of different systems are dissimilar. For example, the geometry may be limited to a few basic types such as points and line strings. Or alternatively, it may include much more sophisticated types such as grids, fields, curved primitives or polyhedrons - in case of 3D models. As with the geometry, spatial (and temporal) referencing may be accomplished in a variety of ways [107].

Differences in syntax are still a major problem area within the exchange of geodata. These issues however, are addressed by many format standardization efforts. The XML-based data transfer standard developed by OGC, GML, should tackle this problem. GML will be discussed in section 3.4.

Semantic differences between geospatial databases occur when there is a disagreement about the meaning, interpretation, or intended use of the related objects [107]. For example, a building in system A can be defined differently in system B. When data from both systems need to be combined, the syntax might be the same, but the meaning of “building” (i.e. the semantics) is not. In addition, the enormous variety of encodings of geospatial semantics makes it particularly challenging to process requests for geospatial information [110].

The semantic diversity of spatial representations causes problems, which can become more complex than the syntactical problems [101], [108]. Solutions for these problems could for instance be found using ontology’s, object classes and/or mediation. Uitermark [106] treats these issues in the context of data set integration. Because this thesis focuses on the technical level of interoperability, these issues will not be discussed further.

When information needs to be shared, several problems with respect to the system architecture may arise [107]:

- Each database management system has its own functionality and interfaces.
- The databases may be installed on different platforms, which support different network protocols.
- The application protocol, which defines the way two or more components (e.g. clients or servers) communicate, may present a problem.

### 3.3. Open systems and GIS Web Services

According to [107], software systems that are built on standards for portability and interoperability are called open systems. [73] demand more than portability and interoperability to call a system an open system:
An open system is a system that implements open interface specifications and standards that promote application portability, scalability, interoperability, diversity, manageability, extensibility, compatibility with legacy components, and user portability.

An open system standard is an interface specification - a specification that describes services provided by a software product - to which any vendor can build products [107]. There are two important points for open system standards. First, the specification is available to any vendor and evolves through a consensus process that is open to the entire industry. Second, the specification defines only an interface; so different vendors can provide the standard interface on their proprietary systems. In addition, the specification must be detailed enough such that two independently developed components can work together based on the specified interface.

3.3.1. GIS Web Services

GIS Web services can be considered a special type of open systems. Web services are self-contained, self-describing, modular applications that can be published, located, and dynamically invoked across the web (ISO, 2001) Web services use the Internet as communications network.

Web services provide access to sets of operations accessible through one or more standardized interfaces. In the process, services may use other external services and operations. GIS Web Services can be grouped into three basic categories [111]:

- **Data services** (such as the OGC Web Mapping, Web Coverage and Web Feature Services) offer customized data to users [112-116]. These services are tightly coupled with specific data sets.

- **Processing services** are not associated with specific datasets. Instead, they provide operations for processing or transforming data in a manner determined by user-specified parameters [117]. Such services can provide generic processing functions such as projection/coordinate conversion, rasterization / vectorization, map overlay, imagery manipulation, or feature detection and imagery classification.

- **Registry / catalogue services** are used to classify, register, describe, search, maintain and access information about Web services [112-116]. Types of registries are differentiated by their role such as registries for cataloguing data types, online data instances, service types and online service instances. The contents of registry / catalogue services are metadata describing other services.

In order for a sustainable and extensible GIS Web Services architecture to exist, the basic services should be accessed via standardized interfaces.

[117] argues that, once GIS Web Services are deployed, client applications can be built more flexibly by mixing and matching available services. Each client application is created by using multiple GIS and non-GIS Web Services like a service to pay for GIS services.

This service-based model is rapidly materializing because of the advancements in general web service technologies and the focused efforts of the Open GIS Consortium (OGC) in the areas of service categorization and interoperability of service interfaces [117] [73]. One of these technologies - Web Feature Services (WFS) – and other components like GML to exchange the data and a spatial DBMS
3.3.2. OpenGIS Service Framework

The OpenGIS Consortium, Inc. (OGC), a not-for-profit trade association dedicated to promoting new technical and commercial approaches to interoperable geoprocessing, was founded in 1994 in response to widespread recognition of the problem of non-interoperability and its many negative ramifications for industry, government, and academia [73]. OGC envisions the full integration of geospatial data and geoprocessing resources into mainstream computing and the widespread use of interoperable, commercial geoprocessing software throughout the global information infrastructure [73].

The OpenGIS Reference Model\(^{11}\) (ORM) provides an architecture framework for the ongoing work of the OGC [102]. Further, the ORM provides a framework for the OGC Technical Baseline. The OGC Technical Baseline consists of the currently approved OpenGIS Specifications as well as for a number of candidate specifications that are currently in progress.

The OpenGIS Service Framework (OSF) identifies services, interfaces and exchange protocols that can be utilized by any application [102] as illustrated in figure 3.1. OpenGIS Services are implementations of services that conform to OpenGIS Implementation Specifications. Compliant applications, called OpenGIS Applications, can then "plug into" the framework to join the operational environment. The OSF is designed to meet the following purposes [102]:

- Provide a framework for coordinated development of new and extended services
- Enable interoperable services through standard interfaces and encodings
- Support publishing, discovery and binding of services through service metadata
- Allow separation of data instances from service instances
- Enable use of a provider's service on another provider's data
- Define a framework that can be implemented in multiple ways

In the OpenGIS Reference Model, some definitions are provided concerning the Service Framework. These key definitions for the Service Framework are:

- A service is a distinct part of the functionality that is provided by an entity through interfaces.
- An interface is a named set of operations that characterize the behaviour of an entity.
- An operation is a specification of a transformation or query that an object may be called to execute.

Each operation has a name and a list of parameters. An instance of a service may be associated with a specific instance of a dataset, or it may be a service that can be used to operate on multiple, unspecified datasets. The first case is referred to as a tightly coupled service, the second as a loosely coupled service. Service operations can be associated with data classes (data type) or with instances (data set).

\(^{11}\) Note that the OpenGIS Reference Model is not a fixed standard, but a living document.

(in the case Oracle has been used) - are subject of this thesis. WFS and other relevant standards developed by the OGC are discussed in section 3.4, and section 3.5.
The architecture is based on the publish/find/bind pattern in figure 3-2 below, and supports the dynamic binding between service providers and requestors since sites and applications are frequently changing in a distributed environment. Three essential roles are distinguished:

- **Service provider**: publishes services to a broker (registry) and delivers services to service requestors.
- **Service requestor**: performs service discovery operations on the service broker to find the service providers it needs and then accesses service providers for provision of the desired service.
- **Service broker**: helps service providers and service requestors to find each other by acting as a registry or clearinghouse of services.

By building applications to common interfaces, each application can be built without a-priori or run-time dependencies on other applications or services. Applications and services can be added, modified, or replaced without affecting other applications.

In addition, operational workflows can be changed on the fly, allowing rapid response to time-critical situations. This loosely coupled standards-based approach to development results in very agile systems - systems that can be flexibly adapted to changing requirements and technologies.
OSF services are accessible from Application Services operating on user terminals (e.g., desktop, notebook, handset, etc.) or servers that have network connectivity [102].

Users may use Application Services to access Registry, Portrayal, Processing and Data Services, depending upon the requirements and designed implementation of the application. Application Services commonly, but not necessarily, provide user-oriented displays of geospatial content and support user interaction at the user terminal. These services can be considered mediating services for users (clients) at one side and primary OSF services (like data services) on the other side.

3.4. Relevant technologies for GIS web services interoperability

In this section, web technologies (e.g., XML) that are defined by the World Wide Web Consortium (W3C) are discussed first. GML (Geography Markup Language) and the OGC Filter Encoding make use of XML and are described next. The last part of this section deals with technologies, which can be used to retrieve and/or edit geographic data through the Internet. These technologies - Web Map Services (WMS), Web Coverage Services (WCS) and Web Feature Services (WFS) – will be compared.

3.4.1. eXtensible Markup Language (XML)

Bos [118] describes XML as:

"XML is a set of rules for designing text formats that let you structure your data. XML is not a programming language and you don't have to be a programmer to use it or learn it. XML makes it easy for a computer to generate data, read data, and ensure that the data structure is unambiguous. XML avoids common pitfalls in language design: it is extensible, platform-independent, and it supports internationalization and localization."

XML is a meta language that allows for creation and formatting of specific document markups [119]. XML can be considered a family of technologies [118]:

"XML 1.0 is the specification that defines what "tags" and "attributes" are beyond XML 1.0, "the XML family" is a growing set of modules that offer useful services to accomplish important and frequently demanded tasks. XLink describes a standard way to add hyperlinks to an XML file."
XPointer is a syntax for pointing to parts of an XML document. An XPointer is a bit like a URL, but instead of pointing to documents on the Web, it points to pieces of data inside an XML file. CSS, the style sheet language, is applicable to XML as it is to HTML. XSL is the advanced language for expressing style sheets. It is based on XSLT, a transformation language used for rearranging, adding and deleting tags and attributes. XML Schemas help developers to precisely define the structures of their own XML-based formats.”

An XML-schema is a document that describes the valid format of an XML dataset. The specification of XML Schema can be found at [120]. Definitions in XML Schema include what elements are (and are not) allowed at any point; what the attributes for any element may be; the number of occurrences of elements, etc.

XSL allows developers to describe transformations using XSL Transformations (XSLT). XSL Transformations can convert XML documents into HTML, another XML-document or other textual output [119]. XSQL Pages are templates that allow anyone familiar with SQL to:

- Assemble dynamic XML "datapages" based on one or more parameterized SQL queries, and
- Transform the "datapage" to produce a result in any desired XML, HTML, or Text-based format using an associated XSLT Transformation.

Using XSQL, one can retrieve XML-encoded data from a relational database by using SQL-statements. It retrieves one feature from a geospatial database (Oracle database, and PostGIS), encapsulates it in XML-tags and transforms it to another text-based format using an XSLT-stylesheet, in this case to an XML-document.

### 3.4.2. Geography Markup Language (GML)

The Geography Markup Language (GML) is an XML encoding for the modeling, transport and storage of geographic information including both the spatial and non-spatial properties of geographic features. GML uses XML-related technologies as well, like XLink or XPointer. The GML specification [122] defines the XML Schema syntax, mechanisms, and conventions that:

- Provide an open, vendor-neutral framework for the definition of geospatial application schemas and objects;
- Allow profiles that support proper subsets of GML framework descriptive capabilities;
- Support the description of geospatial application schemas for specialized domains and information communities;
- Enable the creation and maintenance of linked geographic application schemas and datasets;
- Support the storage and transport of application schemas and data sets;
- Increase the ability of organizations to share geographic application schemas and the information they describe.

Implementers may decide to store geographic information in GML, or they may decide to convert from some other storage format on demand and use GML only for schema and data transport.

A GML application schema is an XML Schema written according to the GML rules for application schemas (i.e., see [122]) and which defines a vocabulary of geographic objects for a particular

---

12 This part is a summary of the GML specification [122].
domain. For example topographic data, cadastral data or road transport data. In such a definition, objects and their application specific properties are defined and references are given to other XML Schemas that define standardized types and elements of GML (see Appendix A for an example of an application schema for GML 2.1.2). For instance, features.xsd is referred to for the definition of geographic features, topology.xsd if topologic relations are modeled or geometryBasic0d1d.xsd and geometryBasic2d.xsd for basic geometric primitives.

An example of a GML application schema for the topographic domain is the recent GML prototype for the Dutch Topographic Service [123]. At the time of writing, the most recent version of GML is GML 3.0. According to the OGC, GML 3.0 provides a variety of kinds of objects for describing geography including features, coordinate reference systems, geometry, topology, time, units of measure and generalized values. Some of these will be discussed shortly. For more details on GML 3.0 see the specification [122].

A geographic feature is an abstraction of a real world phenomenon; it is a geographic feature if it is associated with a location relative to the Earth [124]. This follows the general definition of a feature given in ISO 19109 [125] and the OGC Abstract Specification Topic 5 [126]. The feature XML schema (schemas can be found at http://schemas.opengis.net/gml/) provides a framework for the creation of GML features and feature collections. Associations between features can be defined in GML. One of these associations is ‘features being a member of feature collections’. A GML Feature Collection is a collection of GML feature instances that can behave as a GML feature. GML Feature Collections are themselves valid GML features and can have gml:location and other properties as defined in their GML Application Schema.

With GML 3.0, topology can be encoded. The conceptual model underlying the representation of topology in GML is that of Topic 1 of the OGC Abstract Specification (ISO DIS 19107). The model describes the correspondence of topological and geometric relationships up to three dimensions. One of the requirements in developing of GML 3.0 (and previous versions) is strict separation of data and presentation.

Therefore, none of the GML data description constructs has built-in capability to describe the styling information. In GML 3.0, there is a possibility to describe default styles. The default styling mechanism was created as a separate model that can be “plugged-in” to a GML data set. The term “default” signifies very relaxed relation to other parts of the GML model. The style information that is assigned to a data set may be used for styling but may also be completely ignored. The major advantages of GML representation is that we can build truly interoperable distributed GIS’s.

3.4.3. Filter encoding

A filter expression is a construct used to constraint the property values of an object type for identifying a subset of object instances to be operated upon in some manner. In the Filter Encoding Implementation Specification, an XML encoding for filters is described.

---

13 GML 3.0 has several schemas that define the geometry model. GML 2 has one (geometry.xsd).

14 This part is a summary of the Filter encoding Implementation Specification, of which relevant parts have been adapted.
Filter (a) checks whether the DEPTH is lower than 30, filter (b) combines spatial and non-spatial predicates to check for the geometric property WKB_GEOM to lie within a region defined by a polygon and to check for the DEPTH to be between 400 and 800.

Using the numerous XML tools available today (e.g. XMLSpy), such an XML representation can be easily validated, parsed and then transformed into whatever target language is required to retrieve or modify object instances stored in some object store. For example, an XML encoded filter could be transformed into a WHERE clause for a SQL SELECT statement to fetch data stored in an SQL-based relational database. Similarly, an XML encoded filter expression could be transformed into an XPath or XPointer expression for fetching data from XML documents.

The OGC filter encoding is a common component that can be used by a number of OGC web services. Any service that requires the ability to query objects from a web accessible repository can make use of the XML filter encoding. For example, a Web Feature Service may use the XML filter encoding in a GetFeature operation to define query constraints [127]. Other services based on the Web Feature Service could also make use of the OGC filter encoding.

The OGC filter encoding defines several operators and other predicates that can be used to construct a filter. For example spatial operators (e.g. a Bounding box), comparison operators (e.g. a property is between value A and B), feature identifiers and literals. A literal is any part that is to be used exactly as it is specified. The above Figure 3-3 gives two examples of OGC filter encodings.

Figure 3-3: Two examples of OGC filter encodings

[127]
3.5. OpenGIS Web Services: The WMS, WCS and WFS Interfaces

This section discusses three Web interfaces specifications the Open Geospatial Consortium (OGC) has defined. First, Web Mapping Services (WMS) - to retrieve digital maps over the Internet - are briefly discussed. This is followed by services to retrieve coverages, Web Coverage Services (WCS). Web Feature Services - WFS, which are services to retrieve and edit geographic data are discussed after WCS. This section concludes with a short comparison of these services.

3.5.1. Web Map Service

A Web Map Service (WMS) produces maps of georeferenced data. The WMS specification [18] defines a "map" as a visual representation of geodata; a map is not the data itself. The maps are generally rendered in a pictorial format such as PNG, GIF or JPEG, or occasionally as vector-based graphical elements in Scalable Vector Graphics (SVG) or in Web Computer Graphics Metafile (WebCGM) formats. Although GML is raw data and not a visual representation, GML could be used as output format as well. The WMS-specification standardizes the way in which clients request maps and the way that servers describe their data holdings.

The WMS specification defines three WMS operations: GetCapabilities returns service-level metadata, which is a description of the service's information content and acceptable request parameters in XML. GetMap returns a map image whose geospatial and dimensional parameters are well defined. GetFeatureInfo returns information about particular features shown on a map, by adding to the map. URL additional parameters specifying a location (as an X, Y offset from the upper left corner), the format in which the information should be returned (e.g. as GML, but without the schema) and the number of nearby features about which to return information. This way extra information on geographic features can be requested, but the data cannot be edited.

A standard web browser can ask a Web Map Service to perform these operations simply by submitting requests in the form of Uniform Resource Locators (URLs). WCS and WFS use this technique as well. Figure 3-4 gives an example of a WMS-request to get a map from a server.

A basic WMS classifies its georeferenced information holdings into "Layers" and offers a finite number of predefined "Styles" in which to display those layers.

http://mapserver.somewhere.nl/servlet/wms?
?wMTVER=1.0
& REQUEST=map
& SRS=EPSG:28992
& LAYERS=BUILDINGS,ROADS,RIVERS
& STYLES=default,default,default
& BBOX=160000.0,440000.0,170000.0,450000.0
& WIDTH=600
& HEIGHT=600
& FORMAT=GIF
& TRANSPARENT=TRUE
& SERVICECLASS=someservice

---

15 Summary of [18]
16 Digital representation are meant here
The behaviour of a Web Map Service can be extended to allow user-defined symbolization of feature data instead of named Layers and Styles. Furthermore, individual map Layers can be requested from different Servers. The WMS specification thus enables the creation of a network of distributed Map Servers from which Clients can build customized maps.

A "Cascading Map Server" is a WMS that behaves like a client of other WMS’s and behaves like a WMS to other Clients. For example, a Cascading Map Server can aggregate the contents of several distinct map servers into one service. Furthermore, a Cascading Map Server can perform additional functions such as output format conversion or coordinate transformation on behalf of other servers. In terms of the client-server architecture, a Cascading Map Server is a mediator for Web Map Services.

3.5.2. Web Coverage Service

The Web Coverage Service (WCS) supports electronic interchange of geospatial data as "coverages" - that is, digital geospatial information representing space-varying phenomena [128]. The Web Coverage Service provides available data together with their detailed descriptions; allows complex queries against these data; and returns data with its original semantics (instead of pictures), which can be interpreted, extrapolated, etc. - and not just portrayed. Unlike WFS, which returns discrete geospatial features, the Web Coverage Service returns representations of space-varying phenomena that relate a spatio-temporal domain to a (possibly multidimensional) range of properties.

The Web Coverage Service consists of three operations: GetCapabilities, GetCoverage, and DescribeCoverageType. The GetCapabilities operation returns an XML document describing the service and the data collections from which clients may request coverages. The GetCoverage operation of a Web Coverage Service is normally running after GetCapabilities has determined what queries are allowed and what data are available. The GetCoverage operation returns (i.e., as in figure 3-5) values or properties of geographic locations, bundled in a well-known coverage format.

The GetCoverage syntax and semantics bear some resemblance to the WMS GetMap and WFS GetFeature requests, but several extensions support the retrieval of coverages rather than static maps or discrete features.

```
http://wcs.somewhere.nl/servlet/wcs?
?SERVICE=WCS
& VERSION=1.0.0
& REQUEST=GetCoverage
& COVERAGE=coverageName
& CRS=EPSG:28992
& BBOX=160000.0,440000.0,170000.0,450000.0
& WIDTH=10000
& HEIGHT=10000
& FORMAT=image format
```

Figure 3-5: Example of WCS-request for retrieval of coverage

---

17 This is a summary of [128]
3.5.3. Web Feature Service

The OGC Web Map Service allows a client to overlay map images for display served from multiple Web Map Services on the Internet. In a similar fashion, the OGC Web Feature Service allows a client to retrieve geospatial vector-data encoded in Geography Markup Language (GML) from multiple Web Feature Services.

A drawback of this is that the client has to do the rendering itself. However, besides operations to retrieve data, WFS describes operations to edit (insert, update, delete) data on the level of a feature. With WFS (requests for) transactions on geographic features can directly be sent to a single database. This makes WFS the only open, interoperable standard with which data can be edited over the Internet.

In the specification [129] the Web Feature Service interface is described, as illustrated in figure 3-7. The specification describes the WFS operations. WFS supports INSERT, UPDATE, DELETE (in a Transaction-request) and QUERY (GetFeature) operations on geographic features using HTTP as the distributed computing platform. It also provides operations to describe the service and the contents of the service, such as GetCapabilities and DescribeFeatureType. The specification defines a transaction as a logical unit of work that is composed of one or more data manipulation operations [129].

Transactions can be encoded in XML. It is the function of a Web Feature Service, in its interaction with the data storage system used to persistently store features, to ensure that changes to data are consistent.

However, the document also acknowledges the fact that many systems do support standard concurrent transaction and so proposes optional operations that will allow a Web Feature Service to take advantage of the underlying technologies (e.g. relational database systems). For example, Web Feature Services could support a WFS-transaction for locking (LockFeature).

---

Figure 3-5: Web Feature Service

This is a summary of [129]
3.6. Final remarks

After discussing on distributed computing environment in the previous chapter, this chapter has dealt with a more specific issue of distributed computing environment. Indeed, Interoperable geoprocessing (OpenGIS standards) and GIS Web Services are important subjects in this context. One of these types of services, Web Feature Services, should make editing of geographic data in a distributed, heterogeneous environment possible over the Internet. In addition, the OpenGIS Service Framework provides the framework for OpenGIS Web Services. These open, interoperable services use general web technologies like HTTP, XML and can be used to build applications dynamically. The only services that allow for retrieval of geographic data on the feature level and allow modification of features are services that comply with the Web Feature Services specification.

Web Feature Services are either Basic, which means that data can only be retrieved, or Transactional, which means that the server supports create, update and delete operations on some or all of its features. The WFS specification also defines operations for locking features, which can be used to maintain consistency. WFS requests can be encoded in the URL, using keyword-value-pairs (HTTP, GET, and POST), or in XML. Web Feature Services make use of the Filter Encoding to specify sets of features to operate upon.

To a certain extent Web Map Services, Web Coverage Services and Web Feature Services work similarly. All are interfaces for handling geographical data in web services. However, each of them serves a different purpose. Table 1 compares the most important operations/requests.

<table>
<thead>
<tr>
<th>Operation/request</th>
<th>WMS</th>
<th>WCS</th>
<th>WFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Describe capabilities</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Retrieval of graphics</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Retrieval of unrendered / raw geodata</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Information on a specific feature</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
<tr>
<td>Selection of data is possible by</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bounding Box, Time, Elevation</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bounding Box, Time, values of attributes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spatial and non spatial filters, as in the Filter Encoding</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Editing geodata</td>
<td>-</td>
<td>-</td>
<td>✓</td>
</tr>
</tbody>
</table>

The Web feature service interface uses, GML 2.1.2 to describe the features. This means that the server must support GML 2.1.2 as output format. Servers are allowed to provide any other output format as well. However, GML 2.1.2 is not the most recent version of GML. GML 3 is the latest, which provides encodings for topologically structured data, temporal aspects and default styling as well.
4. Data Integration Concepts

4.1. Introduction

Data integration provides an homogeneous interface to distributed and heterogeneous databases. This homogeneous interface consists of a global (or federated) schema, which is the result of the integration of the schemas of the corresponding local databases. Simply stated, database integration is the process which takes as input a set of databases (local schemas), and produces as output a single unified description (the global schema) of the input schemas and the associated mapping information supporting integrated access to existing data through the integrated schema [130]. Schema integration is a complex and time-consuming problem [131], [132], [133], primarily because the same fact may be contained in several databases yet be represented using different conceptual structures. A main problem in schema integration therefore concerns the detection and resolution of semantic heterogeneity.

The rest of this chapter is organized as follows. Section 4.2 presents a framework for schema integration. Section 4.3 discusses the main issues of schema integration and their possible solutions. In Section 4.4, we describe some of the main approaches schema integration, and finally in section 4.5 we provide concluding remarks to this chapter.

4.2. Framework for Data Integration

For the purpose of this research, this section provides a quick survey of most significant trends in schema integration. To provide a framework for schema integration, we first outline the main steps of a typical methodology and then present the main issues of schema integration. A typical schema integration methodology can be divided into four phases. The steps shown in Figure 3-1 are as follows:

- Pre-integration where local schemas are transformed to make them homogeneous (both syntactically and semantically);
- Correspondence identification devoted to the identification and categorization of interschema relationships;
- Schema integration which solves interschema conflicts;
- Mapping definition involves storing information about the mappings between constructs in the transformed (global/integrated) schema (FCS) and constructs in the local schemas (LCS).

In this section, we take up and develop the example of use cases that has been introduced in Chapter 1. We recall that example comprises two different heterogeneous databases both describing aspects of a municipality cadastral database. Through, we have already illustrated some of the problems of the pre-integration phase (i.e. semantic enrichment and syntactic rewriting). This example will be the starting point for the illustration of the integration processes.
4.2.1. Integration Strategies

In this phase, policies and rules of integration are set. Integration methodologies can be classified as binary or nary mechanisms [134], [134] (Figure 4-2):

- **Binary integration methodologies** involve the manipulation of two schemas at time. These can occur in a stepwise fashion where intermediate schemas are created for integration with subsequent schemas or in a purely binary fashion where each schema is integrated with one other, creating an intermediate schema for integration with other intermediate schemas.

- **Nary integration methodologies** integrate more than two schemas at each iteration. One-pass integration occurs when all schemas are integrated at once, producing the global schema after one iteration.

The binary strategy is simple and efficient, but may lose global information during each intermediate step because not all the information is available at the same time. Although the one-pass integration is complex, it benefits of the availability of complete information about all the databases at integration time. Moreover, there is not implied priority for the integration order of schemas, and the trade-offs, such as the best representation for data items or the most understandable structure, can be made between all schemas rather than between a few.
4.2.2. Pre-integration

In this phase, schemas that correspond to the individual databases being integrated are translated into schemas using a canonical data model [130]. This phase partially corresponds to the logical schemas transformation to make them homogeneous presented further in Chapter 6. We recall that the pre-integration process includes two main tasks, namely, the syntactic rewriting and the semantic enrichment.

**Syntactic rewriting.** Local schemas are translated into a canonical data model. This allows for resolving syntactic heterogeneity this is the result of different data models.

**Semantic enrichment.** This process aims at augmenting the knowledge about the semantics of data. Extracting a semantically rich description from a data source is the main goal of the data reverse engineering process (DBRE). Reverse engineering relies on the analysis of whatever information is available: schema specifications, index definitions, data, and queries in existing programs. Combining these analyses makes it possible to recover hidden structures and constraints [135].

4.2.3. Correspondence identification

The objective of this phase is to identify objects in the underlying schemas that may be related and to categorize the relationships among them. This is done by examining the semantics of the structures in the different databases and identifying relationships based on their semantics. The semantics of an object can be ascertained by analyzing schematic properties of entity types, attributes, and relationships in the schema as well as by interacting with designers and exploiting their knowledge and understanding of the application domain. For example, integrity constraints, cardinality, and domains are properties of attributes that convey their partial semantics. The ultimate objective of this step is the generation of a reliable set of relationships among database constructs. It is important that these relationships be accurate because they are used as input to the schema integration phase.

4.2.4. Schema Integration

In this phase, the interschema relationships generated previously are used to generate an integrated representation of the underlying schemas. Generating such a representation involves resolving various forms of heterogeneity that may exist between related constructs. [136] classifies these heterogeneities into five major categories: domain definition, entity definition, data value, abstraction
level and schematic incompatibilities. The integrated schema generation process resolves these different forms of heterogeneity and generates an integrated schema that hides the heterogeneity from the user.

Taxonomies of conflicts abound in the literature, from very detailed ones [137] to simpler ones [138]. Some examples of well-known conflict categories are:

- **Heterogeneity conflicts**: different data models support the input schemas;
- **Generalization/specialization conflicts**: related databases represent different view points on the same set of objects, resulting in different generalization/specialization hierarchies, with objects distributed according to different classification abstractions;
- **Description conflicts**: the related types have different sets of properties and/or their corresponding properties are described on different ways;
- **Structural conflicts**: the constructs used for describing the related types are different;
- **Data conflicts**: corresponding instances have different values for corresponding properties.

### 4.2.5. Mapping Definition

This step accompanies the integrated schema generation step and involves storing information about the mappings between structures in the transformed (global/integrated) schemas and structures in the (local) wrapper schemas. Such mappings are important for query transformation. It should be noted that these steps may need to be performed iteratively to resolve the heterogeneity and arrive at an integrated representation of the underlying schemas.

### 4.3. Schema Integration Issues

The purpose of this section is to provide a clear picture of what are the main issues and the current solutions in the schema integration field. The focus is on the concepts, the proposed solutions, not on detailed technical discussions. We identify two major issues in schema integration:

- Interschema correspondence;
- Interschema conflict.

#### 4.3.1. Interschema Correspondences

Interschema correspondences are frequently found by looking for similarities in the input schemas. Two local schemas are said to have *something in common* if the real world subsets they represent have some common elements (i.e., a non-empty intersection) or have some elements related to each other in a way that is of interest for future applications [130]. At the instance level, two constructs (occurrence, value, tuple, link, etc.) from two databases are said to correspond to each other if they describe the same world element (object, link or property).

If a correspondence can be defined such that it holds for every construct in an identifiable set (e.g., the population of a type), the correspondence is stated at the schema level. This definition of a
correspondence is called an *Interdatabase Correspondence Assertion* (ICA). The complete integration of existing databases requires an exhaustive identification and processing of all relevant ICA.

In an exhaustive approach, the integration process consists in finding all interschema correspondences and for each correspondence adding to the integrated schema and an integrating description of the related elements (supporting the mapping at the instance level). Local elements with no counterpart are directly integrated in the global schema. At the end of the process, the integrated schema provides a complete and non-redundant description of all data in the global schema. The mappings between the integrated schema and the local schemas support integrated data access for users of the global schema.

### 4.3.1.1. Classification and detection

The objective of this phase is to identify constructs in the local schemas that may be related and to classify the relationships among them. It’s a two-phase process consisting of:

- Identifying constructs that are related;
- Classifying the relationships among constructs.

The first phase requires that the intended semantics of constructs in databases be extracted and constructs that are semantically related be identified. Once a potential set of related constructs has been identified, the second phase involves classifying these relationships into various categories. Roughly speaking, two categories of approaches are distinguished: approaches based on a structural analysis and approaches based on an instance analysis.

**Approaches based on a structural analysis.** Approaches based on a structural analysis use the knowledge conveyed by the various schematic constructs to deduce relationships among constructs. Entity types, attributes and relationships represent the primary schematic constructs that can be analyzed to arrive at these relationships. [139] uses various properties of an attribute to establish relationships among attributes of two different entities belonging to different schemas. The author suggests that attributes be compared on their properties, and it provides definitions for assessing the degree of equivalence of the attributes. For example, entity type could be compared on their names and the description of their roles in the schema. To support the comparison on names, role, etc., sophisticated dictionary and thesauri could be used.

The objective of analyzing these schemas is to identify constructs that are semantically related. However, it is necessary not only to identify but also to classify the relationships among these constructs. The classification generated is dependent on the methodology used. For instance [139], generates four types of equivalences between attributes. There are a EQUAL b, a CONTAINS b, a CONTAINED-IN b, a OVERLAP b. It goes on to define five types of relationships among entities and relationships, each of which can be derived based on attribute equivalences of key attributes. These relationships include A EQUAL B, A CONTAINS B, A CONTAINED-IN B, A OVERLAP B and A DISJOINT B. Users are asked to specify one of these types of relationships for every entity/relationship whose attributes have equivalence relationships specified on them.

**Approaches based on an instance analysis.** The objective of these approaches is to determine instances of entity types in different sources that refer to the same real-world entity. The simplest approach assumes that relations from different databases have a common key. Hence, types that have a common key value identify the same real-world entity [140]. However, a common key may not
always be available. This is referred to as the key equivalence problem. More details can be found in [133] and [141].

4.3.1.2. Resolution

As mentioned before, schema integration consists in determining the interschema correspondence assertions (ICA) by considering both the instance and structural levels. At the instance level, the entity types of the local schemas are usually set into a relationship with respect to the extensional assertions [138]. According to [138], the following binary extensional assertions can be specified between two classes: *disjointness* (≠), *equivalence* (≡), *containment* (⊆), and *overlap* (∩). The disjointness assertion states that the two types are extensionally disjoint in each corresponding database state. The equivalence assertion says that the two types are always extensionally equivalent. The containment assertion is used to describe the fact that one type always extensionally contains the other class. The overlap assertion means that the two types can overlap, that is, they may contain types that refer the same real-world objects.

Types of different local schemas are set into a relationship by a set of schema integration operations. The following list, proposed by [142] and , enumerates the basic operations for "integrating" two entity types ET₁ and ET₂ with the respective intensions¹⁹ I₁ and I₂ and extensions²⁰ C₁ and C₂.

**Generalization:** a new entity type generalizes the entity types ET₁ and ET₂. The intension of the new entity type is determined by the intersection of the intensions of the entity types ET₁ and ET₂. The union of the extensions of the entity types ET₁ and ET₂ gives the extension of the new entity type.

**Specialization:** a new entity type is created as a specialization of the entity types ET₁ and ET₂. The union of the intensions of the entity types ET₁ and ET₂ determines the intension of the new entity type. The extension of the new entity type is given by the intersection of the extensions of the entity types ET₁ and ET₂.

**Subtype:** one entity type becomes the subtype (supertype) of the other entity type in the global schema.

**Merging:** the entity types ET₁ and ET₂ are merged into a new entity type. The union of the intension of the entity types ET₁ and ET₂ determines the intension of the new entity type. The union of the extensions of the entity types ET₁ and ET₂ gives the extension of the new entity type. However, since the new entity, type contains more attributes than the input entity types, default or null values have to be generated for some attributes.

**Partitioning:** the entity types ET₁ and ET₂ are partitioned into entity types with disjoint instance sets. Extensionally, overlapping entity types lead to three entity types (ET₁ \ ET₂), (ET₁ ∩ ET₂), (ET₂ \ ET₁) in the global schema. The entity type (ET₁ \ ET₂) contains all the structures of the entity type ET₁ that

---

¹⁹ The intension of an entity type is determined by the set of structure definitions.

²⁰ The extension of an entity type refers to an actual state of the database at a given time.
are not in ET₂. Analogously, the entity type \((ET₂ \setminus ET₁)\) refers to all structures of the entity type ET₂ that are not in ET₁.

The entity type \((ET₁ \cap ET₂)\) comprises all structures that are in both entity types. The intension of the entity type \((ET₁ \setminus ET₂)\) equals the intension of the entity type ET₁. Analogously, the intension of the entity type \((ET₂ \setminus ET₁)\) equals the intension of the entity type ET₂. The intension of the entity type \((ET₁ \cap ET₂)\) contains the union of the intensions of both entity types. The basic schema integration operations are illustrated in the below Figure 4-3.

**Figure 4-3: Schema integration strategies From [143].**

4.3.2. Interschema Conflicts

When an ICA describes the corresponding types as identical, integration is straightforward. Most frequently, however, the corresponding types can have some discrepancies at both the structural and instance levels. This situation is called a conflict. A detailed taxonomy of conflicts can be found in [136]. In this section, we classify the most common conflicts. Starting with structural conflicts for an overall view, we relate them to instance conflicts and discuss basic techniques for conflict detection.

4.3.2.1. Structural conflicts

Due to heterogeneity at the schema level, schema integration has to deal with various kinds of conflicts. A basic classification of structural conflicts has been introduced in [144]. Following this classification, structural conflicts are classified according two main classes:

- Description conflicts;
- Semantic conflicts.
Description conflicts. The class of description conflicts comprises a large number of more specific conflicts. Here, we only give some examples that illustrate typical description conflicts. Corresponding entity types are often described by different attributes in the local schemas.

This is due to different requirements of the local applications. In one legacy system, local applications need a certain attribute of the entity type whereas in another legacy system, no application requires this attribute. This refers to the missing attribute in similar entity types in different schemas [133]. The solution is to define an abstract type that contains an aggregation of attributes from the underlying types and returns null for those instances that originate from the type that has the missing attribute. However, the query that explicitly retrieves those instances that have null value for the attribute must not retrieve instances from the type that has the missing attribute, since the attribute itself is undefined rather than the values being unavailable.

Other often occurring description conflicts result from the usage of homonymy’s and synonyms for attribute names, entity type names, etc. In general, homonyms and synonyms cannot be resolved in a fully automated way.

Further examples for description conflicts are that corresponding attributes may have different data types or ranges in different legacy systems. Even if they have the same data type, different units of measurement or a different scaling can be using within the legacy systems.

Semantics conflicts. This class of conflicts is caused by the usage of different modelling concepts for expressing the same real-world fact. All data models offer several possibilities to model the same real-world fact. Thereby, local schemas expressed in the same data model can have different structures and constraints although they describe the same real-words facts.

Semantics conflict detection requires knowledge about the problem domain, the local schemas and the extensional correspondences. Thesauri or ontology can support this task, but in general, an automatic detection can only succeed in very restricted cases or application domains.

4.3.2.2. Instance conflicts

This type of conflict occurs at the instance level if corresponding occurrences have conflicting values for corresponding structures. For instance, the same order is stored in two different databases with different customer identification values. Sources for instance conflicts include typing errors, variety of information providers, different versioning, deferred updates [130]. These conflicts are normally found during query processing. The system may just report the conflicts to the user, or might apply some heuristic to determine the appropriate value.

[145] Identifies three kinds of instance conflicts; namely, representation conflicts, identifier equivalence conflicts and attribute value conflicts. For data models with richer expressive power, we could add a further conflict class which refers to relationship conflicts [146].

Representation conflicts: this refers to different representation of data values corresponding to the same real-world fact. This could be caused, e.g., by different unit of measurements (e.g., Belgian Franc vs. Euro), by different notations (e.g., "firstname lastname" vs. "last-name, firstname") or simply different representations (e.g., ISBN with dashes vs. without dashes).
Identifier equivalence conflicts: these arise when instances from different entity types refer to the same real-world object but contain different identifiers.

Attribute value conflicts. They occur when instances, which correspond to the same real world type and share an equivalent identifier, differ in other attributes. One reason for this problem could be a situation, where two entity types from different sources overlap semantically and one of the entity types contains older or outdated data.

4.4. Data Integration Approaches

Data integration consists in providing a uniform view of a set of heterogeneous data sources [147], [148], [149]. This allows users to define their queries without any knowledge on the heterogeneous sources. Data integration systems use the mediation architecture to provide integrated access to multiple data sources. A mediator is a software device that supports a mediation schema (called global schema) which captures user requirements, and a set of mappings between the mediation schema and the distributed sources. Mediation systems [150] can be classified according to the approach used to define the mapping between the data sources and the global schema [34, 151]. The first approach is called global as view (GaV) and requires that each object of the global schema be expressed as a view (i.e. a query) on the data sources [152]. In the other approach, called local as view (LaV), mediation mappings are defined in an opposite way; each object in a given sources is defined as a view on the global schema [153]. Both approaches allow to transform user queries, defined over the mediation schema, into sub-queries defined over the data sources. The transformation process is called a rewriting process [154] and is done differently, depending on the approach used.

After, the short introduction, the rest of the section focuses on the various approaches to data integration and address a set of the main strengths and drawbacks pointed to each approach. Indeed, ontology, schema integration, schema matching and structural data base approaches are provided in section 3.4.1 ontology approach to data integration is shortly discussed, and finally in section 3.4.2 we then overview the schema integration approaches.

4.4.1. Schema Integration

Schema integration is defined as the activity of integrating the schema of existing or proposed databases into a global, unified schema [134]. Schema integration as defined here occurs in two contexts: (1) View integration (in database design) produces a global conceptual description of a proposed database; (2) Database integration (in distributed database management) produces the global schema of a collection databases. This global schema is a virtual view of all databases taken together in a distributed database environment. In addition, the most common solution to data integration and translation is schema integration. Mapping schemas is a technique that takes two data models (or table structures) and produces a matching between elements of the two schemas that correspond semantically to each other [155].

Contrary to the ontologies approach, the process of mapping schemas is based on the content of nodes (or attributes values) instead of a domain knowledge, which uses the semantic similarity between elements (or concepts). This means that the construction of such mapping figures could not be automatically performed without the intervention of specialists in data models or database managers who are closely involved in the content of such schemas.
The advantage of applying mapping schemas in order to achieve the integration of heterogeneous datasets, resides in the straightforward procedure of connecting elements between schemas. This avoids the construction of ontologies for a particular knowledge domain. However, the disadvantage over other approaches is the manual and a time consuming job of linking these components ([156]).

4.4.2. Schema matching

As stated earlier, a fundamental operation in the manipulation of schema information is Match, which takes two schemas as input and produces a mapping between elements of the two schemas that correspond semantically to each other. Match plays a central role in numerous applications, such as web-oriented data integration, electronic commerce, schema integration, schema evolution and migration, application evolution, data warehousing, etc [72]. In all the examples seen so far, an expert user specifies the primitive transformations to be applied on the available schemas and integrate the underlying data sources. In practice, a key issue is the identification of the semantic relationships between the schema object ([148, 157] which then imply which primitive or template transformations should be performed.

The process of discovering semantic relationships between schema objects is called schema matching. Most of the existing methodologies are focused on discovering equivalence relationships between schemas object [42, 148, 157, 158], or direct matches, but in many cases, relationships that are more expressive exist between schema objects, which yield indirect matches.

4.4.2.1. Classification of schema matching approaches

In this section, we classify the major approaches to schema matching. Figure 3.4, shows part of the classification scheme gather with some sample approaches. An implementation of Match may use multiple algorithms or matchers [148]. This allows to select the matchers depending on the application domain and the schema types. Given that, two sub-problems are distinguished. First, there is the realization of individual matchers, each of which computes a mapping based on a single matching criterion. Second, there is the combination of individual matchers, either by using multiple matching of individual name and type equality) within an integrated hybrid matcher or by combining multiple match results produced by different match algorithms within a composite matcher.

For individual matchers, the following largely orthogonal classification criteria are considered [72]:

- **Instance & schema**: matching approaches can consider instance data (i.e., data contents) or only schema-level information.
- **Language & constraint**: a matcher can use a linguistic-based approach (e.g., based on names and textual descriptions of schema elements) or a constraint-based approach (based on keys and relationships).
- **Matching cardinality**: the overall match result may relate one or more elements of one schema to one or more elements of the other, yielding four cases: 1:1, 1: n, n: 1, n: m. In addition, each mapping element may interrelate one or more elements of the two schemas. Furthermore, there may be different match cardinalities at the instance level.
• **Auxiliary information:** most matchers rely not only on the input schema S1 and S2 for example, but also on auxiliary information, such as dictionaries, global schemas, previous matching decisions, and user input.

![Schema Matching Approaches](image)

**Figure 4-4: Classification of schema matching approaches**

[72]

An interesting remark from this classification is that, it does not distinguish between different type schemas (Relational, XML, Object-Oriented, etc) and their internal representation, because algorithms depend mostly on the kind of information they exploit, not on it representation.

### 4.4.3. Modelling (Structural databases integration)

One of the most important aspects in the design of a data integration system is the specification of the correspondence between the data at the sources and those in the global schema. Such a correspondence is modelled through the notion of mapping as introduced in the previous section. It is exactly this correspondence that will determine how the queries posed to the system are answered.

Two basic approaches for specifying the mapping in a data integration system have been proposed in the literature, called local-as-view (LaV), and global-as-view (GaV), known as structural database integration approaches [149, 159].

#### 4.4.3.1. Global-as-view integration (GaV)

The integrated database is virtual. In reality, the local databases are still running on their own. There are no common functions or queries. GaV supports a client driven integration and bottomup development and extension of local source systems. The GaV approach reduces query processing to view processing. Sub-section 4.4.5 provides more detail on this issue.
4.4.3.2. Local-as-view integration (LaV)

[160] and [148], the database integration allows us to build a data warehouse containing all data of the local application. The data of the local application corresponds to virtual or materialized views of the global database. Any change of the local data is harmonized with the global data if the change is going to be supported. LaV supports source driven integration of applications and topondown design of applications by incremental addition of new sources.

In addition, local-as-view integration and global-as-view integration can be expressed through view cooperation expressions [161].

4.4.3.3. View cooperation

[161] Database cooperation is supported by exploiting the import/export facilities of the local databases. Each of the local database systems provides a number of views to the other databases. These views are either export views or import views of the collaborating databases. The schema of an importing view of the importing system contains the schema of an exporting view of the exporting system. View cooperation combines the local-as-view and the global-as-view approaches while maintaining their advantages. The mapping of the databases is similar to LaV mappings. Accordingly, the derivative of the both approaches (GaV & LaV) call both-of-view BaV is based on the use of reversible schema transformation sequences and that BaV transformation can be partially derived from LaV or GaV view definitions.

A key advantage of BAV is that it readily supports the evolution of both local and global schemas [151], allowing transformation pathways and schemas to be incrementally modified as opposed to having to be regenerated. This feature makes BaV very well suited to the needs of peer-to-peer (P2P) data integration, where peers may join or leave the network at any time, or may change their set of local schemas, published schemas, or pathways between schemas[22]. (P2P) data integration approach will be covered on section 4.4.6

4.4.4. Querying in a Data Integration System

Since queries are asked over the mediated schema but the data is stored in the source schemas, queries are posed over the mediated schema and then translated into queries over the source schemas in which the data is stored. For this translation to occur, we need to know how the source schemas are related to the mediated schema. As stated in the above section, two solutions that are commonly proposed in database literature, both of which use the concept of a view – a query have that has been named for reuse. In the first solution, Global-As-View (GaV), the mediated schema is formalized as a set of views over the local sources[148]. In the second solution, Local-As-View (LaV) [34], the local sources are formalized as a set of views over the mediated schema.

4.4.4.1. Comparison between GaV and LaV

The LaV and the GAV approaches are compared in [162] from the point of view of query processing. In GAV, each mediated schema relation is defined in terms of the local sources that correspond to it. Translating a query in GAV requires only replacing the mediated schema relation in the query with the set of local sources that can provide data for that mediated schema relation. Unfortunately, because GAV describes the mediated schema in terms of the local sources, adding new sources is a
complicated procedure; the definition for each mediated schema relation may need to be modified to describe how to retrieve data from the new sources.

LAV, on the other hand, describes each source as a view over the mediated schema. Hence, adding a new source to the data integration system requires only adding new queries describing how the local source can answer queries over the mediated schema relations. Unfortunately answering queries in LAV is much more complex; since the local sources are described in terms of the mediated schema rather than the other way around, translating queries over the mediated schema into queries over the local source schemas is not as simple. In addition, rewriting a query over the mediated schema into queries over the source schema relies on a technique known as answering queries using views; for a survey on answering queries, using views see [149].

Answering queries using views is also used in query optimization – the process of transforming a declarative query into a procedure for executing the query – in order to reduce execution time for traditional database systems. In query optimization, the input is a declarative explanation of what parts of the schema need to be used, and the output is a procedural plan that explains how the query can be executed efficiently. Here views can be used as caches of precomputed information and can speed up the procedural plan considerably, though the underlying relations are available as well.

From the point of view of modeling the data integration system, the GaV approach provides a specification mechanism that has a procedural flavor with respect to the LaV approach. Indeed, while in LaV the designer may concentrate on declaratively specifying the content of the source in terms of the global schema, in GaV, one is forced to specify how to get the data of the global schema by means of queries over the sources. A throughout analysis of the differences/similarities of the two approaches from the point of view of modeling is still missing. A first attempt is reported in [163, 164] where the authors address the problem of checking whether a LaV system can be transformed into a GaV one, and vice-versa. They deal with transformations that are equivalent with respect to query answering, i.e., that enjoy the property that queries posed to the original system have the same answers when posed to the target system.

4.4.5. Peer-to-peer Integration approach

Peer-to-peer systems are distributed systems consisting of interconnected nodes able to self-organize into network topologies which the purpose of sharing resources such as content, CPU cycles, storage and bandwidth, capable of adapting to failure and accommodating transient populations of nodes while maintaining acceptable connectivity and performance, without requiring the intermediation or support of a global centralized server authority.

According to [148, 165], Data integration in peer-to-peer environment use combination of LAV and GaV rules between schemas and a combination of GaV and LAV query processing techniques (BAV). When building an integrated database, one must consider both the logical integration of schemas and their logical extents, and the operational integration of the actual data, defining where the data is to be materialised (i.e. permanent stored) and where data will be virtual (i.e. may be queried, but not permanent stored). Peer-to-peer data integration approach assumes that the logical extent of the local schema equates to the materialised data within such schemas.

Indeed, in P2P network, local schemas are autonomous and membership of the network is likely to be highly dynamic. Thus, maintaining a materialized global schema is likely to be unachievable in
practise, and even answering queries on the global schema is difficult due to the varying nature of local schemas. Hence, a workflow model is mostly proposed as the most promising for development as a basis for P2P integration, but use superpeer schema to make explicit the notion of a global schema that is only implied in the workflow approach [166]. However, the current approach do not assume physical superpeer nodes; rather, it rely on peers publishing via a directory service such as UDDI their integration with standard superpeer schemas that might be owned by any peer.

4.5. Final remarks

Applications often need to integrate information from a variety of distributed and heterogeneous data sources. In order to perform semantic integration of such information, it is necessary to form one or more integrated schemas expressed in some common data model (CDM). Conflicts may exist between the various source schemas, and the schema integration process must remove such conflicts by performing appropriate transformations on the participating schemas. Data integration is possible if different data sources can be combined into a single, integrated format. Therefore, interoperability and standardization of common interface are key factor, in order to support data sharing and data exchange between organizations.

Often full integration is not the aim. The aim is to achieve consistency. In this case the views should be (pair-wise) consistent via some translation mechanism: databases cooperate. This database cooperation mechanism is based on the construction of functions mapping parts of the view instances on parts of the other view instances such as BaV approach. In this chapter, we have presented an overview on data integration different approaches, which subsumes a basic understanding of the issues and of the available solutions. We focused on the fundamental concepts and techniques, insisting on the alternatives and on schema mapping approach based on database views. More details are easily found in the literature.

Several important problems remain to be investigated. Examples are integration of complex objects, n-n correspondences (fragmentation conflicts), and particularly integration of heterogeneous distributed databases (integrity constraints and methods). Theoretical work is needed to assess integration rules and their behaviours related to different domain application. It is therefore important that the effort to solve integration issues be continued and that proposed methodologies be evaluated through experiments with real applications.
5. Core Cadastral Model

5.1. Introduction

Currently in this complex world we live, the importance of standardisation is escalating. Not only to simplify communication within specific domains, but also to generate paradigms and thus preventing that the ‘cycle’ is reinvented over and over again, [167]. Cadastral registration development is one of those things that have been regarded very cultural dependant [168]. Many cadastral systems have their roots far back in history and matured in correspondence with both political and social, needs and resources [169]. This gives the reason for the variety of cadastral systems that exist in the world. While the Dutch and most of the European countries have highly developed, accurate and complex spatial cadastre, African countries such Rwanda and like in most of developing countries have maintained simple land registration, merely to serve fiscal purposes, without storing spatial extent of real properties, mostly stored in analogue format, [170], and [171].

Most countries have in fact their own characteristic land recording system that differentiates for an instance on what is registered (deeds, titles), how it is registered (e.g. fixed- or general boundaries) and where it is maintained (on local or public level) [172]. This gives way for a jungle of systems and information that are hard to interpret or compare against each other. It moreover does not explicitly set an example for other countries to follow in their development. Several contributions have been to the standardisation of the cadastral domain as discussed by [2], on international level, country level and by private companies.

The following chapter in general deals with the CCDM, its goals, structure and potential. In addition, two different data sets are being analyzed as selected data set use case examples from the Dutch cadastral system. It starts with describing the main objectives along giving general overview on the ideology it has with, in section 5.1.1, Followed, is the model presented along with describing in detail it main aspects, covered on section 5.2. Several remarks put forward in section 5.3, especially that concern the potential and criticism of CCDM as observed from literature. Finally, the section 5.4 covered the Use-case data sets analysis based on the global schema, and final concluding remarks are provided on section 5.5

5.2. Standardized cadastral domain model

The Core Cadastral Domain Model (CCDM) was with this in mind brought up at FIG congress in Washington 2002 by Professor Peter van Oosterom (TU Delft, The Netherlands) and Christiaan Lemmen (ITC, The Netherlands), embracing the fundamental concepts of most cadastral systems. Its main purpose is to [169]:

- Enable effective and efficient implementation of flexible (and generic) cadastral information systems based on a model driven architecture, and
• Provide the ‘common ground’ for data exchange between different systems in the cadastral domain.

5.2.1. Objectives

The Core Cadastral Domain Model is defined with the widely used Unified Modelling Language (UML) employing the ideology of model driven approaches, stating that everything can be modelled to UML diagrams describing exact relationships between different objects, their role and flow of information. This facilitates “portability, cross-platform interoperability, platform independence, domain specificity and productivity” of systems [173]. Main objectivities of the CCDM are stated in the opening sentences of [169]:

• A standardized core cadastral domain model, covering land registration and cadastre in a broad sense (multipurpose cadastre); will serve at least two important goals:

• Avoid reinventing and re-implementing the same functionality over and over again, but provide a extensible basis for efficient and effective cadastral system development based on a model driven architecture, and

• Enable involved parties, both within one country and between different countries, to communicate based on the shared ontology implied by the model.

Based on these premises, every cadastre in the world should be included and regarded as extension of the CCDM, being a specialisation of the core model. Furthermore, it was thought that countries without cadastral registration could benefit from using the core to develop their own specific system. Using this model driven approach every cadastral system should consequently be compared through this mutual core, the CCDM. This is for instance illustrated in [43] where the Dutch and the Greek cadastres are queried simultaneously through a **Query Translator** that along with CCDM functions as an intermediate between the two systems.

This approach could later be extended to include all cadastral system from e.g. the countries of the European Union, to enable one central cadastral service, a service that would facilitate cross-border transactions with real properties, accordingly, such approach could work within a country cadastral system that applied a distributed environment in exchanging data with its different municipalities or across the boarder with the neighbouring countries [43]. Particularly, this is an important aspect, in times of heightening international cooperation, establishment of single market within European Union and consequently different environment of real estates transactions [174]. Countries still developing their cadastral registration system could use the CCDM to sharpen their approach to the registration (model driven using UML) while providing core classes that the simply could be picked according to suitability.

5.2.2. Components

Land recording can be described as registering the association between three primary classes, seeing that person is related to land through the rights that he holds. Land is on other hand subjected to right, thus there must at least one right be associated to proprietary land parcel, allowing multiple ownership in accordance do share hold (See Figure 5-1).
The right one is preferred as it allows the same person to own multiple rights in the same property.

The CCDM is in accordance split into three main sections:

- Administrative section, defining how person are registered;
- Legal section, handling the rights and restrictions that are subjected to ownership of real estates; and Geographic section, covering parcel registration and geographic extent.

Together the administrative and legal sections make up what is referred as land registration, while geographic section represent cadastral mapping. The following subchapters present and discuss each section briefly, before the complete model is presented in Appendix C. This summary on the CCDM is mostly based on [15], [174], with assistance of older versions, namely [175] and [169].

5.2.2.1. Administrative section

The administrative part of the CCDM encapsulates the classes that are related to public administration like maintaining a public registry, address repository and legal certification of individuals and companies [15]. This is illustrated in a diagram presented with Figure 5-2.

*Person* is here the centre of attention as that is the class, which links subjects like human beings or companies to property objects through rights or restrictions. There are two principal specialisations of Person: *NaturalPerson* and *NonNaturalPerson*, inheriting Person attributes: `subjectID`, `tmin` and `tmax`. Person is here defined either natural or non-natural with unique identifier, contrary to *Person* other specialisations *Conveyor*, *Surveyor* or *MoneyProvider* that are not assigned unique identifier as they do not create specific instances of the Person class. This is because a *Surveyor* is a *NaturalPerson* but *NaturalPerson* is not necessarily a Surveyor. Thus, the `subjectID` of *Conveyor*, *Surveyor* or *MoneyProvider* always has to match either *NaturalPerson* or *NonNaturalPerson*.

*GroupPerson* intends to represent communities, co-operations and diverse entities that make up social units or structures. An example is e.g. nomadic tribes that ignore modern individualism but live
collective life, grazing the land. Here the tribe can be considered as instance of the GroupPerson whereas each member of the tribe is automatically assigned share of the total.

Figure 5-2: The administrative part of the CCDM

[15]

Some attribute names presented in the diagram can sound unfamiliar and in need of brief definition. BIC is a Bank Identifier Code used internationally to identify financial institutions.

OrgExtID stands for ‘organisation external identification’ and has the purpose of being a link between the CCDM and some kind of company repository. Similarly, personExtID, stands for ‘person external identification’ and provides the link to person repository.

Finally, the variable OID represent ‘object identifier’, an analytical unique identification assigned for objects, e.g. like ‘social security number’ assigned to natural persons in many countries.

### 5.2.2.2. Legal section

In the legal section, the value of the link between RealEstateObject and Person is assigned with an UML link attribute to the class RRR (Rights-or-Restriction-or-Responsibility). RRR has several important attributes [15].

Share is intended to register the amount of the RRR instance that Person has in RealEstateObject. The type attribute has codelist as value that lists all possible types of RRR. \( t_{min} \) and \( t_{max} \), furthermore indicate the temporal begin and end of RRR. Examples of RRR types are e.g. ‘freehold’ and ‘leasehold’. RRR has also several subclasses: Appurtenance, Encumbrance and Obligation that all serve as further specialisation of RRR.

Mortgage, provided by MoneyProvider (lender like bank) is possibly associated to RRR and both refer to a LegalDocument conveyed by Conveyer.

In most jurisdictions there exist also possibilities that properties are affected by negative right, a restriction or advance set by public laws or regulations [15]. Hence, the PublicRestOrAdv reflects how public law influences usage of RealEstateObject, while property law deal with RRR. A PublicRestOrAdv can for instance be restriction like preservation zones made because of natural or
historical relevance, or advance like a building permit as indicated in physical planning. Like RRR the reference document for PublicRestOrAdv is found in a LegalDocument.

5.2.2.3. Geographic section

The geographic section is here presented in two parts. The parcel registry part shows the relationships that the class RealEstateObject has to diverse subclasses. The geography/topological part shows how the spatial dimension is recorded using SurveyPoint and corresponding topology and geometric realisations.

5.2.2.4. Parcel registry

The core of a cadastral registration is the real estate object that is represented in the CCDM with the RealEstateObject class illustrated in Figure 5-4. As an abstract class, no instances can be created of the RealEstateObject, but only at subclass level. NonGeoRealEstate is e.g. intended for properties that cannot be defined geometrically or have not yet been surveyed. Examples of this are e.g. diverse rights that can be separated from land ownership, for example fishing rights in a lake or a river. Similarly are the classes AppartmentUnit, VolumeProperty and RestrictionArea specialisation of RealEstateObject, each registering delimited shapes that real estate objects can take.

All the specialisation of RealEstateObject mentioned until now have in common that they do not partition or cover completely the spatial domain. That does however the role of PartitionParcel.
Parcel along with NPPRegion (Non-Planar Partition Parcel) and ServingParcel partition completely the planar spatial domain and together make up the abstract class PartitionParcel. It is noticeable that Parcel is specialisation of both RealEstateObject and PartitionParcel that could create implementation difficulties, as some program languages and databases do not allow multiple inheritances.

The role of the NPPRegion is to include all PointParcel and SpagettiParcel used to represent spatial locations of parcels that do not have properly registered boundaries. For example, an address location can be used as the initial source for a PointParcel [15]. Based on new data the boundaries could later be indicated by spaghetti data and the parcel moved to the SpagettiParcel class. Finally, when the parcel’s boundaries have been properly measured it is upgraded to be a proper parcel and does not longer belong to the NPPRegion.

ServingParcel compromises two or more Parcel and belongs to (servers) them. Thus, is not registered to Person and not related to RealEstateObject. An example of ServingParcel is e.g. playground or parking area that is jointly used by several parcels.

The Parcel class is the most valued unit of registering land ownership. It is the formal unit of land ownership and subjected to transactions. In this context, both PartOfParcel and ParcelComplex serve as intermediate state of Parcel while splitting or merging land parcels.

Figure 5-5, illustrates how this is implemented with a UML state-diagram. If a part of a parcel is to be sold, the parcel is split into PartOfParcel that get the status of regular parcel when their boundary has been surveyed.
5.2.2.5. Geography/topology

The diagram presented in Figure 5-6 shows how spatial dimension is added to the classes already introduced in the parcel registry part. The basic element here is the SurveyPoint, used by other classes to derive geometry in accordance to ISO/TC211-19107 standard on ‘Geographic Information – Spatial Schema’. A good overview of this standard can be obtained in [176]. Survey points are obtained from ‘survey document’ and the association is that there can exist multiple survey points per document. By using these measured survey point’s classes like:

- **RestrictionArea** and **VolumeProperty** can realise its geometry. For instance, at minimum three survey points are needed to construct two-dimensional RestrictionArea, or at least four points to construct VolumeProperty. Zero to many survey points can indicate ApartmentComplex, used to represent multiple ApartmentUnit, and it is optional if and how it should be represented. Similarly, SpagettiParcel is indicated with at least three points and PointParcel with one.

- **ParcelBoundary** is a combination of two or more SurveyPoint, and has direct geometrical association to instance of the topological TP_Edge. TP_Edge is defined as two end nodes referred here as TP_Node with direct association to SurveyPoint. TP_Edge has furthermore optional number of intermediate points derived from SurveyPoint.

In addition, the geometry of PartitionParcel is constructed from at least one ParcelBoundary, while each ParcelBoundary has relationship to at least two PartitionParcel. This topological relationship is preserved between TP_Edge and TP_Face whereas the geometry of PartitionParcel is realised directly from TP_Face.

Finally, Appendix C represent the complete CCDM as introduced by [177] in Cairo 16-21 April 2005. Looking at all sections together gives more comprehensive view of the model than displaying isolated parts, even though it might seem a bit more complex. Definitions of classes of the CCDM are presented in the Appendix A, to accompany the diagrams. The list of definition shown in that appendix was compiled by using diverse publication of the CCDM [169], [177], [178], and [174] along with a bit of ‘Goggling’.
5.2.3. CCDM, potential and future development

The potential of the CCDM is widely regarded large, but still there is little experience of its utilisation. It is still in its infancy and will need to get through a lot of discussion and criticism before it finally can be regarded as a universal conceptual core for international / cross-platform comparison of cadastres. This section deals with the experience of CCDM so far, along with discussing its main criticism and future development.

5.2.3.1. Experience and Criticism

Hess & de Vries (2004), developed in their work, a query translator, based on the idea of CCDM. The objective was to build a translator that could translate queries between different cadastral systems, in their case, the Dutch and Greek cadastres. The ultimate goal was that a Dutch user could construct a query, valid for the Dutch system, and execute on the Greek system. The query translator would then translate the query appropriately for the Greek system, execute it, retrieve the results and translate it back to the Dutch system. The Dutch user would then receive results similarly as he were querying his own system. This supports the proposed functionality of the CCDM and that it complies with its initial objectives. However, Hess & de Vries found also several limitation of the CCDM that can be used to refine its future version and comparable implementation.

It could considerably increase the complexity of the CCDM if all their recommendations [43] would be realised. For instance, it would require that every national model of the world would be integrated into the CCDM resulting in multiple classes with sophisticated relationships. This is part of the
criticism that the development CCDM has received. As the CCDM is in development and thus subjected to constant revision, external remarks have been stimulated to better the model. This is e.g. reflected in the latest version of the CCDM [177]. One of the main criticisms of the CCDM approach from the Bamberg meeting is found in [179] respectively on page 1 and page 8, where he states:

- The conceptual background of the Core Cadastral Domain Model at the moment needs complex objects to be able to create a correct real property based model. This approach tends to be complex.

- The core cadastral domain model initiative, trying to model existing occurrences of cadastres, is thus confronted in every step with new questions.

Professor Stubkjær [180] from Aalborg University has moreover pointed out that CCDM does not treat transactions and spatial frame reference sufficiently, which he thinks is of significant relevance. The counter argument to this is that spatial reference system should be considered belonging to different repository than cadastral registration. Consequently, this raises the question of the correctness of including person registration within the CCDM.

5.2.3.2. Future development

The Core Cadastral Domain Model is of such nature that constant development and refinement of it are necessary through out its lifetime. While this research is undertaken a new version of it was represented in [181] There it has taken considerable changes from the one presented in this paper, i.e. based on interaction between author and the developers of the CCDM and recommendations from ISO/TC211. Appendix B, presents the newest version of the CCDM diagrams.

5.3. Use-Case data sets analysis

5.3.1. Data sets analysis based on the global schema

As initial phase in this data sets analysis, the definition of attributes for the tables of the core cadastral model should be carried out. In order to provide the three basic tables (Parcel, Right and Person) with attributes, a selection of such attributes has been made from the Cadastre 2014 Data Model ([182]. The three basic tables have been restructured as in Figure 5-7. As explicitly stated in [14] the principles of Cadastre 2014 are integrated with the standardized cadastral domain model, and thus, the proposed direction is fully consistent with the visions and strategies outlined in. The core cadastral model acts as a reference in the integration strategy, and its basic role is to serve as foundation for data structure translation. Based on the data structure presented in Figure 5-7, the next step is to find equivalent attributes from the data sets of the Dutch Cadastral system. Figure 5-8, shows a portion of the UML model Dutch cadastral database-basic classes. The complete model is on Appendix D.
Implementation of the Core Cadastral Domain Model in a Distributed Environment using OpenGIS Standards

Figure 5-7: Attributes assigned to the three main tables of the CCDM

Figure 5-8: A portion of the UML model Dutch cadastral database-basic classes
5.3.2. Local schemas and Global schema attributes definition

As stated previously, two data sets from the Dutch Cadastral System are being analyzed. The data set from the Municipality of Gouda has been chosen. This data set is taken as a sample of the model implemented in the rest of the country. The data structure of the core cadastral model is being matched with the structure of the selected data sets based on their attributes. Table 5-1 shows the first table of the core cadastral model (Parcel) and identifies the corresponding tables within the data set. In addition, a composition of field-to-field (or field-to-fields) associations is made with the attributes of the tables.

5.3.2.1. Person association

Table 5-1: the following considerations are based on the Person table association

<table>
<thead>
<tr>
<th>Dutch</th>
<th>CCDM</th>
<th>Dutch to CCDM</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBJECT_ID</td>
<td>subjid</td>
<td>subjid</td>
</tr>
<tr>
<td>ONTVANGSTDATUM</td>
<td>tmin</td>
<td>tmin</td>
</tr>
<tr>
<td>VERLIJDENSTDATUM</td>
<td>tmax</td>
<td>tmax</td>
</tr>
</tbody>
</table>

ParcelId or objectid (after mapping). X_AKR_OBJECTNUMMER is a unique code for the identification of land parcels in the Dutch Cadastral System, therefore, ‘X_AKR_OBJECTNUMMER’ is the primary key for the tables related to land parcels in both data sets (parcel geometry data as well as administrative data). It is supposed that there are no duplicates due to the composition of this code. It is a string character, composed of eleven (116) digits (i.e., GDA01M04047G0000). The first three digits correspond to the code of department (or district, in this example this will be from “Gouda Municipality”, e.g., “GDA01”). Each department is politically divided into several municipalities (or districts). The subsequent two digits are then, the code of the municipality (e.g., “M04047”). The next two digits represent the code of a subdivision created for cadastral purposes called ‘polygon’ (or sector). Finally, the last five digits stand for parcel number (e.g., “G0000”).

Right association

Table 5-2: the following considerations are based on the Right association table

<table>
<thead>
<tr>
<th>Dutch</th>
<th>CCDM</th>
<th>Dutch to CCDM</th>
</tr>
</thead>
<tbody>
<tr>
<td>AANDELL</td>
<td>Share</td>
<td>Share</td>
</tr>
<tr>
<td>SOORT_RECHT</td>
<td>RightType</td>
<td>rightType</td>
</tr>
<tr>
<td>X_AKR_OBJECTNUMMER</td>
<td>ParcelId</td>
<td>objectid</td>
</tr>
<tr>
<td>GERECHTIGDE</td>
<td>PersonId</td>
<td>subjid</td>
</tr>
<tr>
<td>ONTVANGSTDATUM</td>
<td>tmin</td>
<td>tmin</td>
</tr>
<tr>
<td>VERLIJDENSTDATUM</td>
<td>tmax</td>
<td>tmax</td>
</tr>
</tbody>
</table>

It is also, important to note that each attribute from the tables of the data sets (Dutch cadastral system), are being translated according their main meaning. Indeed, for the integration strategy, the purpose of these attributes is not strictly taken into consideration, but the meaning. It is not the
intention of this research to design the most advantageous database model for the Dutch case, because the existing system works nicely. The true intention is to provide a valuable insight into the complexity of the integration of heterogeneous data sets using a standard data model. Database experts could further readapt the proposed data structure for specific purposes.

5.3.2.2. Parcel association

Table 5-3: the following considerations are based on the Parcel table association

<table>
<thead>
<tr>
<th>Dutch</th>
<th>CCDM</th>
<th>Dutch to CCDM</th>
</tr>
</thead>
<tbody>
<tr>
<td>X_AKR_OBJECTNUMMER</td>
<td>ParcelId</td>
<td>objectid</td>
</tr>
<tr>
<td>GROOTTE</td>
<td>estimatedsize</td>
<td>estimatedsize</td>
</tr>
<tr>
<td>PP_1_NR</td>
<td>partnum</td>
<td>partnum</td>
</tr>
<tr>
<td>GROOTTE</td>
<td>legalsize</td>
<td>legalsize</td>
</tr>
<tr>
<td>ONTVANGSTDATUM</td>
<td>tmin</td>
<td>tmin</td>
</tr>
<tr>
<td>X_AKR_OBJECTNUMMER</td>
<td>Parcelname</td>
<td>parcelname</td>
</tr>
</tbody>
</table>

According to these three associations, when designing databases, it is advisable to rely integrity rules and constraints on its database management system (DBMS) instead of dealing with programming routines for validations. The construction of such “X_AKR_OBJECTNUMMER” code demands a considerable attention in order to restrict invalid values being entered to the database.

Furthermore, the inclusion of municipality and district codes explicitly implied on its composition may represent further problems if the case of changes in municipality borders arises (i.e., the same case applies, if two municipalities have to be merged as one municipality). The problems coming along with re-codification tasks or primary keys modification are highly expensive.

Therefore, it is recommended to analyze carefully the implication of storing the Cadastral (Id) or Code in such way that takes into consideration the numerous advantages of managing integrity constraints offers by means of a database management system (DBMS).

5.4. Final Remarks

Nowadays, the Core Cadastral Domain Model innovation is much-needed contribution to the cadastral discussion and general development [181]. It has an ambitious future vision and could benefit both developed countries, for cross-platform interoperability, and undeveloped countries, giving them core to build on own sophisticated cadastral system, avoiding reinventing the wheel again. It is however, the view of this paper that it is hard to fulfil both its objectives of setting an example for effective cadastral system development and create shared ontology to enable cross-system interoperability. Not mentioning conditions stated in [181]:

*Important conditions during the design of the model were and still are: should cover the common aspects of cadastral registrations all over the world, should be based on the conceptual framework of Cadastre 2014, should follow the international ISO and OGC standards, and at the same time the model should be as simple as possible in order to be useful in practise.*
It is the feeling here that the intention of creating a universal applicable cadastral core model should be segregated from the objective of creating an example for cadastral systems to follow. Argument for this is that a paradigm requires a conceptual model, which is brought to implementation level within each country. Cross-platform interoperability however should be more concerned with the technical implementation, using physical model.

Covering common aspects of cadastral registration all over the world creates a very complex model, not easily interpretable by few others than its developers. In addition, the question can be asked why producing complex universal applicable CCDM, covering various heterogeneous indigenous variations like nomadic rights, when such model does not yet exist for Europe, with more homogenous legal and cultural background.

Separating the implementation of “cadastral system paradigm” and CCDM could: (Encourage discussion on a cadastral paradigm much like Cadastre 2014, but using more explicit tools like UML to express its future vision and objectives; and Give the CCDM more room to evolve and grow technically to become universal applicable as a common denominator of diverse cadastral system).
6. Integration Architecture

6.1. Introduction

Information systems that provide interoperation and varying degrees of integration among multiple databases have been termed multidatabase systems [183, 184], federated databases [185, 186], and more generally, Heterogeneous Distributed Database Systems (HDDBS) [187]. A mechanism to achieve that goal is through an integrated schema that involves all or parts of the component schemas. The taxonomy presented by [41] classifies the existing solutions in three categories: global schema integration, federated databases, and multidatabase language approach. These categories are presented according to how tightly integrated component systems are.

The chapter is organized as follows. First, we present a taxonomy that classifies the existing database integration architectures solutions into three categories: global schema integration, federated databases and multidatabase language approach. Second, we describe the approach used for integrating cadastral database architecture. Thirdly, we propose relevant interoperable standards and open interfaces in order to integrate distributed computing.

6.2. Overview of integration architecture

As stated earlier, depending on the level of integration, integration architectures can be classified into three categories: global schema systems, multidatabase language systems and federated databases. These categories reflect design efforts to accommodate the conflicting requirements of achieving efficient and usable system by larger level of sharing on one side, and preserving the autonomy of the data sources, on the other.

On the one extreme of this spectrum are the systems that are close to the distributed databases in building a global integrated schema of all the data in the sources. The opposite side represents systems that provide just basic interoperation capability and leave most of the integration problem to the user. The rest of the section overviews the features of each of these categories.

6.2.1. Global schema systems

A straightforward approach to building an heterogeneous distributed database system (HDDBS) is the approach where the export schemas of multiple databases are integrated into a single view (global) schema [188]. In [134], a thorough survey on schema integration is provided and twelve methodologies are compared.

The user is not aware of the distribution and the heterogeneity of the integrated data sources. Multiple databases logically appear as one single database to users. Furthermore, if the schema does not change frequently, database can be stored locally, at the client, for faster access. Nevertheless, this approach has been shown to exhibit the following problems [187]:

---

73

---
Since the general problem of integrating even only two schemas is undecidable, the process of integration of multiple schemas is very hard to automate. Global schema integrators must be familiar with all the naming and structure conventions of all the data sources and integrate them into a cohesive single schema without changing the local schemas.

There are two basic approaches to integrating the component schemas into a global schema. In the first, the component schemas are integrated pair-wise. A hierarchical application of the integration leads to a schema integrating all component schemas. The other approach is to integrate all the component schemas at once. Both approaches have problems. The first one could produce different results when different integration orders are used, while the other one is usually too difficult (Cfr. Chapter four, these approaches are explained in depth).

The global schema integration is not suitable for frequent dynamic changes of schemas, as the whole process of integration may need to be redone. As a result, it does not scale well with the size of the database networks.

### 6.2.2. Multi-databases language

This approach does not provide any type of global schema. The only means of accessing the data in the data sources is by language primitives for specification of queries over data stored in multiple sources. Information stored in different sources may be redundant, heterogeneous and inconsistent. These problems occur when components system are strongly autonomous.

The aim of a multidatabase language is to provide accesses involving several databases at the same time. Such language has features that are not supported in traditional languages (i.e. database query language). For instance, a global name can be used to identify a collection of databases. Queries can specify data from any local participating database (example: MSQL [184]).

The main criticism of the multidatabase language approach is the low level of transparency provided to the user. The user is responsible for finding the relevant information, understanding each database schema, detecting and resolving the semantic conflicts, and finally, building the required view of the data in the sources. The advantages of the approach are that it is not intrusive against the autonomy of the data sources and there is no global/federated schema maintenance and access overhead.

### 6.2.3. Federated Architecture

In the federated Multi Database Management System (FDBS), the export schemas are only a subset of the component schemas. The federated schema does not need to be an integration of all the export schemas. It can be integrated only portions of the export schemas of interest to the users using the federated schema. More than one federated schema can be defined according to user’s requirements. Each user can then further refine its export schema to fit his own requirements.

The aim of this architecture is to remove the need for static schema integration. It allows each local database to have more control over its sharable information. It should be noted that FDBS is a compromise between no integration and total integration. A typical FDBS architecture would have a
common data model and an internal command language. The level of integration and services in a FDBS depends on how tightly/loosely coupled the component DMS\textsuperscript{21} are.

6.2.3.1. Tightly coupled systems

In a tightly coupled FDBS, federation administrators have full control on the creation and maintenance of federated schemas and access to export schemas. The aim is to provide location, replication and distribution transparency. This approach supports one or more federated schemas [65].

A federation repository keeps the mappings between the different schemas and helps maintain uniformity in the semantic interpretation of multiple integrated components of data. The size of the repository can grow dramatically as the number of data sources and users increase. It can also become a performance bottleneck when accessed by a large number of users. These problems are reminiscent of the problems of maintaining a global schema described above.

Once a federated schema is created, it is rarely changed; that is, it is static. This, it does not support dynamic changes of export/component schemas.

6.2.3.2. Loosely coupled systems

Loosely coupled systems do not have a centralized administrator. The user creates and maintains his own integrated schema in the form of a local view. Creating a federated schema corresponds to creating a view against the relevant export schemas. In that respect, each user must be knowledgeable about the information and structure of the relevant export schemas in order to create views [27]. Federated schemas here are dynamic and, as a result, can be created and dropped on the fly. Multiple federation schemas are supported. The maintenance problems noted above disappears.

A possible drawback of this approach is that more than one user might need to perform the same view modelling, without the possibility of reusing the definitions. Furthermore, a change in an export schema affects all the users who have a view dependent on it.

A solution to the problems noted above is to allow a gradual transition from the federated into export schemas by a hierarchy of small intermediate schemas. This approach breaks the repository into smaller and more maintainable units, while allowing reuse of the view specification and modularity in the view definition and change [189].

6.2.3.3. Mediation architecture

To address the problem of interoperability of information systems in general, the term mediation has been defined in this context [11] as a service that links data resources and application programs. The function of a mediator is to provide integrated information, without the need to integrate the data resources. A mediator hides details about the location and representation of relevant data to applications. Several prototype mediator systems have been developed [150], [51] Figure 6-1 shows the basic architecture of information processing using mediators. Compared to the client/server architecture, mediation introduces an additional layer in the architecture of information systems.

\textsuperscript{21} DMS standing for Data Management System, a term encompassing both File system and Database systems
6.2.3.4. Mediation and databases

A database federation offers a virtual, integrated view of the underlying component databases. Queries issued against this view are translated into queries against the underlying component databases. A database federation performs mediation using at least two important components [190]: wrappers and mediators. The function of a mediator is to provide integrated information, without the need to integrate the data resources. A mediator hides details about the location and representation of relevant data to applications.

Above each database is a wrapper. A wrapper is a software component that performs the translation between the export schema and the physical schema of a source [191]. That is, the wrapper (1) offers an export schema in the canonical data model (2), accepts queries against the export schema and translates them into queries understandable by the underlying database, and (3) transforms the results of the local queries into a format understood by the application.

Wrappers and mediators rely on schemas descriptions and mappings to translate queries and to form the result instances.

6.3. Database integration

The current methodologies developed for building a data integration are generally based on a database integration approach e.g.,[192-194]. Referring to [193], the database integration is made up of four main processes: (1) preparing the database schemas; (2) integrating them; (3) defining the mappings and (4) building the architecture components.

6.3.1. Schema integration principle

The proposed methodology for schema integration is based on mapping schemas. Mapping schema is a technique that takes two data models (table structures) and produces a matching between element of two schema that correspond semantically to each other [31, 195].

6.3.1.1. Principle of mapping schemas

The mapping process involves the analysis, design, and maintenance of mapping between pairs of schemas. A mapping process is a set of operations applied to a source schema to translate it into the form of a target schema [196, 197] (i.e., in our use case the source part of the Dutch cadastral data
model) schema are translated into mediated schema - the core domain model (CCDM). The fundamental principle behind the mapping schema technique is to find the correspondences between the attributes of a pair of schema. In order to fulfil this, the use of function to combine, transform or manipulate multiple fields is vital and any overlapping or schema conflicts have to be identified and solved.

### 6.3.1.2. Conflict identification

In the database integration using schema integration methodology, conflicts can occur in three possible ways: syntactic, semantic and instance.

**Syntactic conflict**, besides the usual conflicts related to synonyms and homonyms, a syntactic conflict occurs when different object types in local schemas represent the same concept.

**Example** For instance, the concept *OrderDetail* can be represented by an attribute (Figure 6-2, site 1) or by an entity type (Figure 6-2, site 2).

![Figure 6-2: Syntactic conflict examples](image)

*The concept OrdDetail is a multivalued attribute in the site 1 whereas it is represented by an entity in the site 2.*

**Semantic conflict**, a semantic conflict appears when a contradiction appears between two representations A and B of the same application domain concept or between two integrity constraints. Solving such conflicts uses reconciliation techniques, generally based on the identification of set-theoretic relationships between these representations: A = B, A in B, A and B in AB, etc.

**Example**: Figure 6-3 shows an example of a semantic conflict between two attributes. In the site 1, the attribute Phone is optional (its minimal cardinality is 0) whereas the same attribute is mandatory in the site 2.
Figure 6-3: Semantic conflict example. The attribute Phone is optional in site 1 & mandatory in site 2

Instance conflicts, instance conflicts are specific to existing data. Though their schemas agree, the instances of the databases may conflict. This problem has been discussed in [13]. This process is highly knowledge-based and cannot be performed automatically.

Example: Consider the two schemas of the part of Figure 6-4. Common knowledge suggests that USER be a subtype of EMPLOYEE. However, data analysis shows that inst(EMPLOYEE) \subseteq inst(USER), where inst(A) denotes the set of instances of data type A.

6.3.1.3. Conflict resolution

In chapter 4, we state that most conflicts can be solved through four main techniques that are used to rework the local schemas before their integration: renaming, generalizing, transforming and discarding.

1. **Renaming**, Constructs that denote the same application domain concepts are given the same name.

2. **Generalizing**, If constructs denote the same application domain concept, and if one of them is more constrained, the constraint is relaxed. For example, a [0-10] cardinality conflicts with a [1-N] cardinality. Both will be replaced with cardinality [0-N] is the strongest constraint compatible with both source cardinalities.

3. **Transforming**, An application domain concept can be represented by constructs of different nature in source schemas. A supplier can be represented by an entity type in schema 1 and by an attribute in schema 2. The latter construct will be transformed into an entity type to give both representations the same nature.

4. **Discarding**, A construct that conflicts with others can be merely ignored. This is the case when the former appears to be a wrong translation of the application domain concept.
6.3.1.4. Local Source Schemas merging

Since the syntactic, semantic and instance conflicts have been resolved by restructuring the local schemas, merging the latter is straightforward, and can be automated largely. Note that conflict resolution need not be completed as a preliminary process. Indeed, conflicts can be completely or partially solved when merging schemas. According to this strategy, the source schemas are left unchanged, and merging each pair of (sets of) constructs can imply on the fly restructuring in order to solve conflicts.

6.3.1.5. Building the inter-schema mappings

As stated in chapter 4, our approach to data integration is schema transformation oriented in that we focus on providing mechanism for defining schema correspondences between each local schema and the global schema (Figure 6-5) and, on then using that equivalence to define the mediator mappings in form of database views or sources views. As a result, transformation sequences for each pair of local and global schema have to be defined.

From the previous discussion, it is clear that schema integration and schema-mapping development are complex and time-consuming processes and automation is desirable. However, automation of the process presents number of challenges. [198] notes that the schema integration process cannot be completely automated. This is primarily because two same schemas can be integrated differently, based on their intended use [137]. It is however possible to reduce the amount of human interaction [133].
Figure 6-5: Schema integration as a set of schema transformation sequences ($T_1$ and $T_2$).

No tool has yet been developed as a commercial product. Some research projects have produced significant prototypes. They are dedicated either to the integration process or to the building of mediators:

- **For building the integration process:** [199, 200] and (Hayne, 1992) propose tools for automated inter-schema relationships identification. [142] presents a set of tools that support different issues of the process, e.g., methodology, conflict and similarities identification, semantic extraction.

- **For the building of mediator (federation components):** [191] proposes an implementation toolkit that facilitates the rapid development of mediators; HERMES [201] provides a set of tools to support the construction of mediators.

Many of these tools, however, appear to be limited in scope, and are generally dedicated to a limited aspect of the federation development. They do not attempt to integrate techniques and reasoning common to the integration process and the building of data integration architecture components, leaving the question of a general tool and architecture for developing an open data integration system unanswered. For the purpose of this research, the graphic image (Figure 6-6), is made up in Protege + OWLViz (ontology editor and visualization plug-in), in order to show the mapping at entity type level (class level), which gives the overview picture of the mapping process.

### 6.3.2. Analysis of the schema-mapping process

#### 6.3.2.1. Mapping the Dutch model to the core domain model

As stated in the previous section, mapping schema is a technique that reconciles two data models and produces a matching between elements of two schemas that correspond semantically to each other. The approach supposes a matching between two data models. This may be considered far from true, parting of the fact that, in practice software tools nowadays such as Mapforce\(^\text{22}\) and StylusStudio\(^\text{23}\) only permit matching between two tables (or schemas). The addition of more than one source table is feasible; however, only one target table is permitted. In addition, data models are composed of not only object types (tables) but also relationships and regulations. Hence, the limitation of mapping between only a pair of structures does not allow preserving such relationships.

---

\(^{22}\) [http://www.altova.com/products_mapforce.html](http://www.altova.com/products_mapforce.html)

Therefore, in order to get all the structures of the sources schemas and the global schema, we selected and proposed to work with the database integration approach, which is done by creating views (For The actual implementation of the mapping in the "create view" Cfr. Chapter 7) of different sources database according to the global schema (mediated schema). Simply, stated as illustration the following is a “create view” statement of the mapping at attribute level (how to go from the Dutch attributes to the CCDM attributes for each class). Figure 6-7, provides a clear example of the underlying approach to create views. It shows how the mapping are done at the database level in order to integrate two data sets (i.e. here precisely are on (a) the (mapping) integration between the RegisterParcel class from the Dutch model into the core, and (b) the mapping between the Right Class from the Dutch model into the core domain model - CCDM).
Figure 6-7: Create view statement for data integration

In (a) is a Create View statement to map the RegisterParcel class from the Dutch model into the Core domain model (CCDM) and in (b) is a Create View statement to map the Right Class from the Dutch model into the Core domain model.

6.4. Architecture for data integration

Traditional data integration approaches, such as data warehousing, are costly to deploy and do not easily support either real-time access to data or dealing with autonomous sources. For the purpose of this research here, an alternative solution is proposed. Indeed, an architecture for data integration in the case of cadastral system sector, which facilitate distributed query, is outlined in this section. Firstly, the use-case requirements that have to be fulfilled in order to design such architecture are identified and discussed in the next sub-section.

6.4.1. Data integration Architecture Requirements

The following requirements have been derived from the user case preconditions to design such architecture for data integration and distributed query.

6.4.1.1. User requirements

The design architecture should be to help the end user of the system to be able to get an integrated product from the data exchange between both sides of the system.

Integrated product

- Map, Legal data

Quality check

- Aggregated products (i.e. compute the average price by making use of the Zip Code area), and access based on postal address.

Technical requirement

The system boundaries of the core model are stricter than those of the national cadastral models. Classes from 'outside' the cadastral domain, e.g. (Postal) Address, are not incorporated in the core
model. Especially if the core model has to be adopted by developing country such as Rwanda, the class person should be within the system boundary, this mean that, it would be nice if the system keeps not only the person name but also, the sex and the Date of birth. Therefore, the following requirements must be fulfil and respected in the database structure model:

- There must be link between object Id and address
- There must be Consistency and key constraint on Unique Id’s should be required

### 6.4.1.2. Web client Requirements

Functionalities necessary or critical for the client to test the concept of data integration and distributed query are discussed here. Other functionality, for instance concerning a good user interface, is left aside in the case study, because this is not critical to test the main goal of distributed query execution.

Only necessary functionality for the use case has been implemented. The use case geospatial features data contains polygons and points. Thus, the client will not deal with creating polygons. The client for this case study should be able to:

- Connect to a web feature server (preferably more than one) and optionally other services, like WMS
- Support the retrieval of GML for all data (GetFeature);
- Support selection queries based on the non-spatial attributes of features (parcels, rights, persons, documents etc.);
- Be generic as possible for flexibility of testing. The table of contents and the selection forms for specifying the queries should be dynamic and not ‘hard coded’ (static);
- Visualize GML;
- Identify GML feature, i.e. give the thematic attributes of a feature;
- Send the WFS-request to the server and interpret the responses.

Section 7.2.2 discusses the implemented client.

### 6.4.1.3. Requirements for the Web Sever

The physical layer of the architecture is comprised of web services that receive the request from the clients, receive the data from the data sources, and pass the data on to the client. These web services “hide” the implementation details of the data sources from the client because they can be accessed using a standard Open GIS protocol. Easily configurable as wrappers to access cadastral databases as long as data format is concern. The server is configured on top of the PostgreSQL/PostGIS 8.1 spatial database. In terms of WFS, the server should be able to:

- Interpret GetFeature requests to make corresponding SQL-statements and encode the result of the SQL in GML for sending it back to the client,
- Support filters to select features on the feature identifier and a spatial filter to select features in a bounding box,
- Make valid WFS responses for error reporting.
- Provide XML-schema per feature type in response to a DescribeFeatureType Request.
6.4.2. Architecture components

The architecture is based on two emerging paradigms: Web services\(^{24}\) and Enterprise Information Integration (EII). Web services are rapidly becoming the de facto standard for interoperability between different software applications, running on a variety of platforms and/or frameworks. EII systems are based on wrapper-mediator architecture [202]. In this approach, the data from the distributed sources is not transferred to a new central repository. Instead, data remains at the source and the EII system is responsible for providing users with virtual unified views over the source data. When the mediator receives a query, it decomposes it into sub-queries over the sources of data, executes them in real time and integrates the sub-query results to obtain a global query answer. The proposed architecture based upon EII wrappers and Web services is shown in Figure 6-8 in more generic way and shown in Figure 6-9 in details with all its main components.

In this architecture, the possible sources of data include internal databases, proprietary applications, systems from partner or member companies accessible through Web services interfaces, and autonomous Web sites from external organizations (specialized content providers, competitor’s Web sites, and so forth).

![Generic data integration architecture](image)

**Figure 6-8: Generic data integration architecture**

The **physical layer** of the architecture is comprised of Web services (Wrappers) that make sources conform to a common model and which is easily configurable (i.e. Web Feature Service) to access cadastral databases through mediated schema (or global schema) for that domain (e.g. for the cadastral domain), see Figure 6-9.

The **integration layer** allows creating unified "virtual" views over data from the sources stored in PostgreSQL/PostGIS and executing queries written in a database query language (such as XQuery and/or SQL). Thus, users are able to obtain answers to queries very similar to the ones they could issue against a hypothetical conventional database containing all the information from the sources. Since sources can be semi-structured or even no-structured, the system should also support combining database-like queries with text-based searches. However, this would mean chaining a series of WFS requests in a particular WFS client so to say (Note that, service chaining is a different topic).

\(^{24}\) Web Services Activity at the W3C; [www.w3.org/2002/ws/](http://www.w3.org/2002/ws/)
Although, this is an interesting research, it is very complex and it requires much more time. Therefore, this was left out of our objectives.

Additionally, the access to the system will typically be made through a Web service-client interface or a simple geo-spatial Web Client such as uDig [203].

**Client**, in the Web client the data from the cadastral geo-database, data serves as background for integrating administrative and parcels geometry data. With a spatial query (based on the current bounding box) also other data, like orthophotos or topographic data from other WFS or WMS services, can be added in the client as additional background information.

The WFS service should make it possible for users to access the sources database over the Internet as illustrated in Figure 6.9. The client should consist of a viewer and some layer that “talk” WFS, i.e. transforms operations from the viewer to valid WFS requests and handles communication of requests and responses with the Web feature server.

![Detailed data Integration Architecture](image)

**Figure 6-9: Detailed data Integration Architecture**

The open source WFS server GeoServer (http://geoserver.sourceforge.net) has been used in the use case. GeoServer is a full implementation of the WFS specification of the Open Geospatial Consortium. GeoServer can be configured as a Web Feature server on several data formats, including PostgreSQL\PostGIS Spatial database, (the cadastral data used in this case study is stored in a PostGIS database).

The developed client uses SVG (SVG, 2004) for the visualization of the GML output of the WFS service. SVG can be generated by transforming the GML output stream with an XSLT (XSLT, 2004) stylesheet. For more detail on the current functionalities of the Web client is described in section 7.2.2.
6.5. Interoperability

A basic principle for interoperable services, in this case WFS, is that any WFS client should be able to communicate with any WFS service. In our use-case, we used the prototype client to access two WFS services, i.e. a WFS service set up at TU Delft with cadastral data from a different municipality as the one mentioned in the use-case section (See, Figure 6-10 as illustration).

Besides the use-case Web client, we tested another WFS client: uDig Viewer. However, with uDig client, it was not possible to perform selection queries by specifying filter conditions. Meanwhile, the fact that features from an 'unknown' data source can be retrieved, visualized and queried by just any WFS compliant Web client shows the power of interoperable Web Feature Services.

![Figure 6-10: Web feature Services interoperability for data integration](image)

6.6. Concluding remarks

This chapter have tried to synthesise all the previous chapters in order to get an overview of our research approach to data integration.

The current approach presents a framework that applies a set of interoperable components based for data integration. We have proposed a geospatial non-materialized data integration system architecture, which extends existing data integration systems to accommodate specific GIS problem (i.e., cadastral domain data integration). The system complies with standard mediation /wrapper architecture described in the literature, although most of existing systems/prototypes do not deal with spatial data.

The implementation we provide in order to identify requirements for such architecture is based on a use-case related to cadastre land registry data integration example. The underlying architecture complies with OGC standards and specifications such as GML and derivative XML, and WFS as interoperable standards suitable for such architecture. We extend previous approach (data integration, at the databases level) with new concept of derived wrapper, which contains the sources schema translation defined by using database views in the database layer, and which is used to cope with a specific GIS issue in this case, cadastral domain data integration. Concepts in the core model, which are present as equivalent or specialized concepts in every national cadastral system, can therefore be exchanged between these cadastral data sources.
The next chapter discussed our approach in a more technical way by creating views of the underlying conforming model as a mediated schema (global schema), that serves as an open interface to distributed and heterogeneous data sources.
7. Proof of concepts

7.1. Web Feature Servers

At first the plan in this research was to identify available suitable interoperable standards, which could facilitate the integration of geospatial cadastral data. The first alternative was to develop a WFS client, but this was involving to create GML application Schema for the available data, and run the XSLT in the PostGIS database. Since developing a Web Feature Server with XSLT would have consumed lots of time here was decided to investigate the use of Web Feature Servers as GeoServer [204], released implementation from the Open Source projects.

GeoServer claimed to be full compliant as basic and transactional web feature server, i.e. for all feature types all operations were said to be supported. Therefore, GeoServer was investigated further and used to develop the use case prototype service.

This chapter is covering; the technical part of this research which deals with the proof of concept related to the implementation of the underlying data integration architecture and distributed query testing.

7.1.1. GeoServer an Open Source Web Feature Service

GeoServer is a full implementation of the WFS Specification of the OpenGIS consortium. It is developed in Java and supports versions 1.3.0 RC-6, and 1.1.0 (the current version) of the Web Feature Service specification. GeoServer version 1.3.0 RC-6 has been extensively tested against the official OGC test suite [204].

GeoServer can be configured as a transactional Web feature server and is originally developed for PostGIS25. Nevertheless, GeoServer is developed to be able to deal with other data sources as well. At this moment, GeoServer can act as a Basic WFS on ESRI’s shapefiles and GeoServer could be used on an Oracle Spatial database and ESRI’s ArcSDE as a Transactional WFS. The fact that GeoServer supports PostGIS is relevant, as PostgreSQL/PostGIS is also the platform that we selected as our database package, since PostgreSQL, is an object-relational database that has the geospatial extension enabled, and as we were interesting to work with full Open Source packages.

Because GeoServer already had implemented all spatial filters (as it is a full implementation of the WFS-specification), GeoServer 1.3.0 RC-6 was chosen for further development of the Web Feature Service. GeoServer 1.3.0 RC-6 has been configured to serve as a Web services (WFS) on top of the cadastral database stored in PostgreSQL 8.1 (we recall the main architecture Figure 6-9). Appendix E gives some details on configuring GeoServer on top of a PostgreSQL database.

25 PostGIS is an open source spatial database technology that adds support for “OpenGIS style” geographic objects to the PostgreSQL object-relational database (PostGIS, 2005)
Furthermore, GeoServer 1.3.0 RC-6 contained some bugs that troubled especially processing transactions. However, with this current version we have been lucky. In this research, most of the functionalities expected from GeoServer were supported correctly at least as basic WFS. Besides that, the only thing that is not convenient is that Join between features types of different WFS are not supported yet in the current Web Feature Services OGIS Specifications and hence, not supported yet supported in GeoServer.

Therefore, the need for a specific client was required, which will make use of WFS current protocol, and enhanced them in order to test distributed query.

7.1.2. Existing WFS clients

Commercial GIS software are not able to connect to PostGIS, which results with that only open-source GIS clients qualify as the front end to geospatial data. There are several clients available and easily customisable to the needs that claim to be WFS compliant have been tested however, none of them qualifies for the identified architecture requirements to cadastral data integration, especially for distributed query environment e.g. Jump, QGIS and uDig. The applicability of each of this clients is here discussed in more detail.

Jump 1.1.2 is right away ruled out of, as it does not support connection to PostGIS based on PostgreSQL version higher than 7.4. The client is however extremely user friendly and gives a real GIS feeling when querying, styling or editing data. The future of the client is unclear. The source code is being used in the new uDig client while some devoted Jump users have continued developing Jump under the name OpenJump (http://openjump.org).

The only real option left is the uDig 1.0.5 client, which besides its many limitations is probably the desktop client currently available that can integrate the most variety of data sources. It can access several SDBMS (i.e., PostgreSQL/PostGIS); edit diverse vector formats, and WFS, plus being capable of viewing WMS. It has a drawback to be incredible slow and computer memory consuming.

The main advantage of uDig apart from it ability to display and edit diverse spatial data is its ability to be extended and customised according to users wish, as long as he has knowledge of Java. Being open-source software the source code is available for download making the possibility to create customised functions, e.g. specially intended for cadastral or surveying processes. Finally, as it is partially based on the Jump-Project, as the initial objective of the developers were to combine the functionality of the two open-source projects Jump and GeoTools. This could indicate that future versions of uDig will develop in the direction of Jump with all its GIS functionalities. Figure 7-1 illustrates the main objective of uDig to become cross platform desktop GIS client.

Another advantage is how uDig transacts with PostGIS, using conventional SQL transactions. it has a major drawback of not performing queries as expected. Moreover, demand quite an effort to customise this interface in order to perform even simple selection queries.

The conclusion of the above is that, although many clients (websites) are available for web mapping, for Web feature services and in particular, a full transactional Web feature services that could successfully perform distributed query, not much of such clients are available yet (At the website of
OpenGIS (http://www.opengis.org/resources/?page=products) conforming products are listed). Therefore, it is decided to develop a generic transactional WFS client capable to perform distributed query, especially for the purpose of the use-case of this research.

![Diagram of spatial data systems](image)

**Figure 7-1:** in uDig, documentation states that “…
Interactive desktop access is the missing application in the open-source OpenGIS standards-based spatial infrastructures”[203]

### 7.1.3. Developed Client for the Use-case

The developed client is based on a client made by the Section GIS Technology (TU Delft) that has been used to visualize GML-encoded data from Top10NL as SVG (Scalable Vector Graphics) in a web browser [205] and [206] This SVG-client is adapted to use a Web Feature Service as data source. The client has been extended with functionality to draw and encode features and send these in a WFS-transaction to the server. The prototype is at present built solely for testing, thus the user interface itself is very straightforward. With it however we can test several scenario (selection queries) and hence evaluate the success and also limitation of our proposed architecture for data integration and our approach to distributed query environment.

Both the GetCapabilities and the GetFeature request (for retrieval of the features in GML) are sending as Keyword-Value-Pair-requests (GET, POST, and HTTP) and are issued using Javascript. If SVG is chosen to visualize the data, XSLT is used in the browser to do the transformation from GML to SVG. A screenshot of the client in Figure 7-4 shows some features in SVG. The transformation to SVG is done at the client side, after receiving the GML.

For navigation (zooming and panning) the client uses standard SVG-functionality. Support for SVG has to be available in the browser; a plug-in (e.g. from Adobe) can be used for this. Note that not for all web browsers such a plug-in is available, for instance for Internet Explorer. The thematic attributes of features can be queried by clicking on a feature. A table with the attributes is generated from the GML in the client (see, Figure 7-2).

The part of the client that is responsible for the retrieval and visualization of GML is generic in the sense that all Web Feature Services could be used as a data source. For the use-case, support for distributed query execution had to be implemented however.

Processing is partially done at the server, the results are returned to the browser. This way, functionality of the WFS-client is distributed over the browser and a web-server. Note that the
machine that has been used for the client and the web feature server are in fact on the same computer.

Functionality to compose and send query requests is divided between browser and web server, (it works just like uDig: it sends WFS requests to a WFS service (i.e., Geoserver - WFS) and accepts responses of that WFS service), as follows:

- Specifying the service: type in the URL and choose 'capabilities' the client sends a GetCapabilities request to the WFS service, and when the service responds (with a GetCapabilities XML stream) this XML stream is transformed by a XSL stylesheet into the Table of Contents (left frame).

- Select a feature to query: choose the feature types (by checking the check box) and do GetGML another request is sent to the WFS service (GetFeature request) and you get the GML back (WFS response).

When the feature type has geometry (like 'parcel' and 'gebouw') you can also draw it on the map (by clicking the button). Additionally, by clicking on a geo-object in the map, again a GetFeature request is sent to the WFS service, but now only for that object, using the object gml:id. At this level, this is the basic functionality, just as uDig (see, Figure). What is extra (not in uDig), is that we can make selection queries by specifying filter conditions.

In addition, the user can send the Transaction request from the browser to WFS server (GeoServer). This will send the (raw) XML-request to the web feature server and passes the response (successful or failed) of the server to the browser.

**Figure 7-2: The developed WFS client interface**
### 7.2. Database integration

According to the use-case requirements, we have proposed a client-server architecture, see Figure 6-9. This architecture allows the integration to be done in a flexible and open manner, which allows sharing components and data between (sub) systems. Data views have been the ability to combine two or more physical databases and make them look like one integrated database.

Because of the distributed environment of the cadastral data, a system such a query tool must be open, i.e. it cannot operate on a local vendor specific data format. Most of the current GIS-systems do not support this open database approach. The traditional approach is that data has to read into the GIS which store it in a local proprietary format. However, there are GIS’s, which means that the geometry information still has to be stored locally, but with help of a key, the corresponding administrative data can be found. This is still not good enough, since complete integration of geometry and administrative data is the most efficient and consistent data management architecture.

Therefore, an approach to database integration such as create database views from the mediated schema translation of the data sources, make services of these views through the WFS was preferable. The following section demonstrate the proof of concept in apply the underlying approach.

#### 7.2.1. Answering query using database views for data integration

##### 7.2.1.1. Create view definition

Talking about integration of data in a relational database system environment means talking about integration of data models. Each data model represents a part of the real world, and is maintained by a specific application. Therefore, the data model may not be changed. Integration is done at the level of data views and is implemented by using the relational join mechanism.

The integration model can be realized with the following view:

The above Figure 7-2, illustrates how the create view syntax is defined in SQL statement.

<table>
<thead>
<tr>
<th>Command:</th>
<th>CREATE VIEW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description:</td>
<td>define a new view</td>
</tr>
<tr>
<td>Syntax:</td>
<td>CREATE [ OR REPLACE ] [ TEMP</td>
</tr>
</tbody>
</table>

**Figure 7-3: Create view general syntax definition**

The following is an example of one of our mappings translations processes which are done at the database level in order to conciliate the different data sources with the global schema (CCDM). In this example a view “parcel” is created with the selected attributes accordingly to the matching between the table or class parcel in the core model and the table xfio_parcel which represent the class parcel in the Dutch cadastral data set (see, Figure 7-4).
7.2.1.2. Use case examples

We described a number of test cases, from simple queries to more complicated ones.

An example of a simple scenario is the following case: a class in the data source has another name may be other names for attributes as a class in the core model, but is intended as the same concept: core:NaturalPerson versus Dutch:NAT_PERS_CODE

An example of a more complicated scenario is a query that involves an association between two or more classes. This would mean – in relational database terms – a join between two tables:

```
SELECT name, address, type_of_right from naturalperson, right
WHERE person.id = right.person_id
AND municipality = '....'
```

Let say, if the same person appeared to have two different parcel in two different municipalities. Here the complicated factor is not the different names for classes and/or attributes, but different (names for) association plus knowledge about the join attributes (foreign key) that must be used.

In this second case it might not be straightforward to rewrite the query based on the schema translation formalized in the mediated schema (global schema). An alternative solution would to use the person’s date of birth, because it may happen that the person_id in the database in one municipality is different to the second municipality, here would be duplication of the same person attributes in represented differently in two databases. The idea also, would be to use the national person social id number kept in the national registry.

7.2.2. Use case architecture solutions

Another approach to solve the problem that data sources models are closer to implementation then the core model would be to make the sources domain models used in the translated mediated schema more conceptual and therefore to have e.g. one n:m association, instead of two 1:n associations. This means that the mediated schema has to be defined in a wrapper application that has to map from one conceptual model to the other and from the conceptual model to its technical data model.

Technically, this could be handled by the by a Query translator [43], but could be also part of the Web service application software. This last architecture set-up would move the responsibility for correct mapping/translation between the conceptual and the technical model to the Web service provider, and leave it out of the ‘middle layer’ to which the query translator belongs in the overall architecture [43].
At the database level, a second approach to solve the problem that data sources views are created across databases and create a WFS service of the resulting mediated/global view which would be query from a common client interface (i.e. see Figure 7-3). Currently this solution is entirely possible using Oracle database package. It is probably possible to create a view across multiple PostgreSQL/PostGIS databases. The bottleneck is to configure the access to the databases that are involved: for each database with tables in the view it is necessary to configure the access rights by specifying a range of IP addresses that have access to it from other computers.

![Diagram](image_url)

**Figure 7-5: Approach to data integration at the database layer**
Create view database approach across distributed databases systems

### 7.3. Open Source and its applications

Evolving trend in software development is the so-called open-source licensed applications as a counterpart to the more commercial closed-source and proprietary developed products. This chapter starts with defining what open-source licences are before covering the commercial usage of open-source products. After, the following Section, discusses geo applications provided under open-source, followed by a section covering their potential for cadastral applications. Finally, the last section concludes the chapter with some remarks.

#### 7.3.1. Open-source by definition

The definition of open-source is a group of licenses obeying to the open source protocol that states that the source code of a computer software should be made available free of charge, for both modification and redistribution (OpenSource, 2005).

Examples of open source licenses are like Berkeley Software Licence, Common Public Licence, GNU General Public Licence (GPL), GNU Lesser General Public Licence (LGPL), to name just a few.
Of these mentioned licenses, the GPL is far the most used open source licence. The difference between them can be manifold, e.g. does LGPL allow the software in demand to be bundled with other software with different open source licence or even proprietary, which is not the case with GPL. The main difference between GPL and BSL is that BSL allows derived work to be redistributed as proprietary software, which is not the case under the GPL licence [207].

There is little conceptual difference between open source licences and those that are developed under the free software movement (should not be confused with freeware that is free but closed source) licences that emphasise more on giving the user freedom. Some examples are found where open source software cannot be categorised as free software and vice versa [207] e.g., some open source software licences allow third party to modify a code, redistribute, and charge as its own.

7.3.2. Commercial usage

The idea of free software is often related in peoples mind to limitations of usage or exploitation. This applies e.g. to freeware, shareware or demo software where: usage is restricted to personal uses; people are required to buy the software if they enjoy using it (appealing to normal conscience); has restricted usability compared to a full-version edition which is charged for; or is restricted to limited trial period with some of the functionality disabled. None of this however applies for open source licensed software, as it would violate the set protocols mentioned before.

Open-source software is meant to be free of charge and with full functionality, independent of user or intended usage. Freely available to governmental institutions fulfilling their tasks and duties, and private companies with commercial purposes. Moreover, it is possible to modify and customise the source code of the software to adjust or extend its functionality for the proposed tasks.

The general conclusion is that open-source applications currently provide the tools to implement “open-source cadastre database” although refinements on user interface, documentation and sometimes software stability is often needed. The availability of open-source software for geo-applications is the topic of next section.

7.3.3. Open-source geo-applications

It is characteristic of open-source software that they reflect international standards, specifications and trends, better than their commercial counterparts do. Spatial features and methods are in principle in accordance to OGC Simple Feature Model developing to the more advanced ISO-19107 Spatial Schema [208]. They support GML as export, offer WMS, WFS and even WFS-T services.

The variety of available solution is vast and the discussion below will only reveal the tip of the iceberg of open-source solution currently available. Not only are new applications added regularly while other drop out of development, but new versions of older applications can enhance their functionality to such an extent making them incommensurable to predecessors. For more information, the main gateway to open-source GIS applications is www.freegis.org.

This section will provide brief overview of the open-source applications that could benefit the implementation of cadastral database. The discussion will start with spatial databases and GIS applications, before moving to spatial data servers, ending with clients and interfaces.
7.3.4. Spatial Database Management Systems and GIS Applications

Two main options for open-source licensed spatial databases exist currently, PostGIS / PostgreSQL and MySQL. PostGIS on PostgreSQL is the most popular open-source spatial database solution. PostgreSQL is actually an OR-DBMS with only limited support for spatial types [209]. However, as the object factor enables freedom for additional customised data types and functions to be declared, the ground was made for the development of PostGIS (PostGIS, webpage).

- PostGIS adds support for geographic objects to the PostgreSQL object-relational database. In effect, PostGIS “spatially enables” the PostgreSQL server, allowing it to be used as a backend spatial database for GIS’s, much like ArcSDE or Oracle’s spatial extension.
- PostGIS is further based on the open-source application program interface referred as Java Topology Suit (JTS) that provides (JTS 2005, webpage).
- Powerful procedural programming language, PL/pgSQL is integrated in the PostgreSQL database making it easy to create customised functions and triggers. It also supports variety of other procedural languages as e.g. C++, Python and Perl (PostgreSQL 2005).

MySQL on the other hand supports spatial data types as defined in OGC Simple Features by default without needing additional module like PostGIS. This is however new feature in MySQL and still in transition. No recent comparison to PostgreSQL/PostGIS could be found, but mid-year 2003 comparison by Refraction Research, the developers of PostGIS, indicated that the spatial capabilities of MySQL was somewhat faulty but promising. Comparing the recent MySQL 5.1 with PostGIS documentation reveals that not much difference is between the two systems, implementation wise, as both are quite loyal to the OGC Simple Feature model.

Only one open-source GIS application is available that is comparable to the GIS paradigm set by commercial developers like ESRI, Intergraph and MapInfo. GRASS, an acronym for ‘Geographic Resources Analysis Support System’, is described as “…a raster/vector GIS, image processing system, and graphics production system” [210]. Its spatial data management capabilities are further stated to include: spatial analysis; map generation; data visualization (2D, 2.5D and 3D); data generation through modelling (list of simulation models); link to DBMS (PostgreSQL, others via ODBC); and, data storage.

The main advantage of GRASS over other GIS systems, beside obvious capital savings of buying licenses, is its interoperability to other open-source products like PostGIS/PostgreSQL. The main disadvantage is on the other hand its complexity and dependency on Unix/Linux and UNIX like environment, making it difficult for Windows users adapt to it, let alone compile it [211].

---

[26] This paragraph was summarized from [211]
8. Conclusions

As more and more data is stored by more and more people or organisations, the number of overlapping sources of data about any given topic is going to increase. In that milieu, it will be increasingly important to have systems that are capable of rapidly querying multiple databases simultaneously in order to integrate and exchange data in a distributed environment.

In addition, the technological and standardisation advancement and increasing user’s demand of customised GIS products have pressurised Geo-Service providers to deliver real-time based distributed GIServices. The existing GIS web services focus on the technical part of problems concerning interoperable geoprocessing. The international standards organisation like, Open Geospatial Consortium has developed several GIS Web Services, based on ICT standards of the W3C, like XML and HTTP. In this context, Web Feature Services are unique type of services, since they are the only open standards-based services that can be used for editing and retrieving geographic data. However, they are not yet capable of sharing computational resources including flexibility in execution of distributed query environment. Therefore, there is a need to explore the current interoperable standards specifications to address these performance problems. The proposed data integration architecture based on a cadastral domain use case requirements, attempts to provide the flexibility in querying distributed heterogeneous sources and integrate data by exploiting the current interoperable standards/technology and Open-sources applications.

8.1. Research questions revisited

The following paragraphs provide the general conclusions derived from the research. The questions posed in section 1.5 are specifically outlined and critically discussed.

Research question 1: What are the suitable interoperable standards and the feasible methods for integrating distributed heterogeneous data sets. What have to be extended?

This question has been addressed in chapter two up chapter six and the sub-question has been answered in chapter seven.

The current interoperable standards such as GML are suitable and mature enough to be used as interoperable open standards especially for the geospatial data exchange. First, they support data integration for instance by allowing a flexibility and resolution of different data format and data representation conflicts. In addition, the use case prototype presented in this research, shows that the retrieval and combination of geo-data from multiple, heterogeneous data sources in one Web client is relatively easy with OpenGIS WFS services, one reason is that the WFS specification clearly describes the request and responses that a WFS service should support. This way is possible to ‘decouple’ client and server and e.g. build an application-specific Web client that still can communicate with Open source or commercial) server software developed by others. Another reason is that WFS uses standard web technologies as HTTP and XML (GML and WFS-request/responses). Not only Web clients, but also thicker clients (like GIS software that has more possibilities for
analysing the data) can use data from WFS servers. This makes exchanging and sharing geo-spatial a lot easier: whether the data is stored in a local file system or in a remote database has become (almost) transparent to the end-user.

However, even though, the current OpenGIS standards presents the underlying advantages, still the OGC WFS protocol is not simple and even complex queries are not a problem (i.e. a simple “Join”) that is easy to do in a local environment (within one database for example) is not so trivial in a Web services context. In theory there is one other mechanism that could work (apart from chaining request in a sequence, one after the other). This is the mechanism to use DescribeFeatureType response (in XML Schema format) to specify the relations (associations) between feature types, and use those relations in the client to help the user construct a valid WFS GetFeature request that can then be sent to the WFS service.

**Research question 2:** *What are the use case requirements to be fulfil such architecture for data integration?*

This question has been solved addressed in chapter six. The preconditions and requirements required to fulfil the system architecture design for data integration, has been provided based on the use case requirements, those are the user requirements and technical requirement from the user point of view.

Also, requirements from technology and data format, the requirements for the necessary functionalities for the client, provide requirement for the WFS client, and the physical layer of the architecture functionalities which is the bridge between the clients receiving the data from the sources through databases wrappers, these provides the Web Server requirements.

**Research question 3:** *How can be used the core cadastral domain model as basis for data exchange and integration, in order to facilitate query execution in a distributed environment?*

This question has been discussed, partially in chapter 4 as global schema data integration approach and particularly in chapter 5 that is entirely dedicated to the CCDM. It has an ambitious future vision and could benefit both developed countries, for cross-platform interoperability, and undeveloped countries, giving them core to build on own sophisticated cadastral system, avoiding reinventing the wheel again. Based on these premises, every cadastre in the world should be included and regarded as extension of the CCDM, being a specialisation of the core model.

This make easy communication based on the shared ontology implies in the model. For example, in our approach we have take advantage of that quality, and proposed the CCDM as the mediated schema for the two cadastral data sources used in the use case.

Nevertheless, the view of this report is that this innovative (CCDM) is important but may be little bit too ambitious in its approach. There is not necessarily shared ontology across cadastral systems as experienced by Hess & Vries (2004) and by including the variation of all cadastral systems tends to make the model complex in contradiction to its objectives. Alternative would be to focus its development more within cultural homogeneous area like Europe before extending it to other regimes. Also it is considered here beneficial to sharpen it approach by identifying the core packages and emphasise their development separately.

**Research question 4:** *What is a suitable approach to data integration and distributed query to fulfil the identified requirements?*
This question has been covered in chapter 4 and fully in chapter 7 as proof of concept. In chapter four we provide thorough and deep study on the different approaches for data integration, for the purpose of this research, and according to the use-case requirements architecture for cadastral data integration, it was relevant for us to make use of the proposed CCDM by FIG, as it encapsulate the basic worldwide cadastral model. By adopting our approach to schema translation and defined local schema in term of the core model (CCDM) it gives much and better facility to integrate cadastral data, the use case example implementation shows that the combination of the underlying approach with the current OGC open interfaces such as Web feature services, facilitate the integration of distributed heterogeneous remote data.

Finally, the methods that were followed and somehow evaluated (especially the OpenGIS standards) in this research, they can provide basis testing distributed query using the available interoperable standards and incorporate even more standards developed by the international organizations (OGC, ISO, W3C and FIG). But, distributed query especially on heterogeneous database remains unsolved, especially in this “Open fashion” many efforts have to be made in future research (some are recommended in this research) which will help to enhance the currents interoperable standards.

### 8.2. Recommendations

#### 8.2.1.1. Future work

As we stressed previously, currently, there are not many (open source) WFS clients which can execute distributed queries, and the one's which exist cannot perform selection queries. Therefore, the GIS technology department TU Delft has built the generic WFS client which we tried to extend and applied for doing WFS distributed query testing in this research.

Accordingly, what is ‘future work’ is to find a solution for the joins (because this would mean chaining a series of WFS requests in order to join different services automatically). Indeed, in our approach, with the current version of the underlying WFS client, we were able to execute join queries “on the fly” in an non automatically way so to speak (many operations were done manually), therefore efforts has to be made in order to provide the end-user with a graphical user interface for such prototype.

Another recommendation for future work is to experiment the so-called global view across multiple databases and combined it with the interoperable open interface specified by OGC.

Indeed, in our approach we translate and applied the mapping schema translation by matching the sources schema to the core cadastral model. Similarly, it could be more relevant to create a view across multiple PostgreSQL/PostGIS databases and services that global view in WFS which will query as a mediator against user query. The bottleneck is to configure the access to the databases that are involved: for each database with tables in the view it is necessary to configure the access rights by specifying a range of IP addresses that have access to it from other computers.

#### 8.2.1.2. Core Cadastral Domain Model

More research is needed combining people from Land registration and Cadastre; they are domain expert and check if the current CCDM provide the needed functionality.
Indeed, although the CCDM is a much needed contribution to the cadastral development and ongoing debate. It sets an example for modelling and development of cadastral systems, while at the same time defining the core classes of cadastral registration, enabling cross-system interoperability. The view of this report is that this innovative is important but maybe little bit too ambitious in its approach. There is not necessarily shared ontology across cadastral systems as experienced by Hess & Vries (2004) and by including the variation of all cadastral systems tends to make the model complex in contradiction to its objectives. Alternative would be to focus its development more within cultural homogeneous area like Europe before extending it to other regimes. Also it is considered here beneficial to sharpen it approach by identifying the core packages and emphasise their development separately.

8.3. Contribution of work

The academic contribution of this project is diverse. In short it can be stated that:

- For sure, it will influence the next version of the CCDM that will includes some of the recommendations put forward here, with the author listed as one of CCDM co-authors.
- The report further gives overview and shows the potential of open-source software available at present, with spatial extension enabled and implemented successfully in PostGIS.

Finally, this research constitutes a viable method for integration of distributed heterogeneous data sets based on schema translation using database views in the database layer, combining with the filter encoding capability in the WFS used as databases wrappers to filter the selection query from the WFS client to the data sources. Therefore our approach contribute as attempt to solution of distributed query issue of the current OGC WFS protocol, and propose a more realistic approach by combining both underlying approaches (e.g. The database level and the OGC standards level).

8.4. Discussion and experience

There are several reasons why this research drifted from its initial objectives (reference is given here to our first proposal for making the caste study to Rwandan scenario).

First of all, it would be difficulties of re-engineering the Rwandan Land administration system model, with no useable references to be found, which consequently implies that the system is developed with a bottom-up approach. Enormous time would be spent trying to model and understand the Rwanda cadastral system. Accordingly, our main objective was to see in which way the Core Cadastral Domain Model (CCDM) and open-source applications benefit the development of nationwide cadastral system in Rwanda.

From the experience I got with this research using the CCDM, it’s clear for me that the development of the Rwandan cadastral registration can greatly benefit from the CCDM, both in general and in implementing spatial delimitation of parcels. The CCDM further offers a conceptual framework and health model driven architecture approach to the whole development. Employing CCDM would moreover bring the Rwandan cadastral registration on level of international cooperation of cadastral research and development, which could proof invaluable in the long run.
The role of open-source application in the development and designing such architecture could be diverse. The research report argues that several open-source applications available are serious candidates in developing spatial enabled cadastral registration system. Examples of applications are PostgreSQL with PostGIS to store spatial data in SDBMS; MapServer on Apache HTTP server to share spatial data in a web environment; standalone GeoServer enabling WFS-T to spatial data; and finally uDig as a desktop client that can be extended with diverse customised functionalities by accessing and editing the open source code.

However, before employing open-source applications for the underlying Cadastre system development, few things have to be taken into consideration. Most of open-source clients are not complete coming out of the box but would need additional tuning and customisation to suit in a cadastral development. Another thing is that there is in most case no one responsible manufacture liable for the correctness and reliability of the application, and the product could even drop out of development without warning. This is because development of open-source applications is largely depended on the enthusiasm and drive of the open-source community. This result in the maxim that if open-source application are to be used, one should handle them with certain reservation and above all be an active member of the open-source community developing the product.
Bibliography


[58] GiMoDig Geospatial info-mobility service by real-time data-integration and generalisation, IST-project No. IST-2000-30090. (date of access)


Implementation of the Core Cadastral Domain Model in a Distributed Environment using OpenGIS Standards


Implementation of the Core Cadastral Domain Model in a Distributed Environment using OpenGIS Standards


Implementation of the Core Cadastral Domain Model in a Distributed Environment using OpenGIS Standards


Implementation of the Core Cadastral Domain Model in a Distributed Environment using OpenGIS Standards


[209] Gardner, J. Materialized Views in PostgreSQL. (date of access, [accessed, Jan. 05] at)


## Appendix A: CCDM definition classes

<table>
<thead>
<tr>
<th>Classes</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdminParcelSet</td>
<td>Aggregation of many parcels that form an administrative unit, e.g. municipality, county, province and election district.</td>
</tr>
<tr>
<td>Advantage</td>
<td>E.g. building permit, the right to build a house on a certain parcel.</td>
</tr>
<tr>
<td>ApartmentComplex</td>
<td>Composition of several ApartmentUnits that form one complex.</td>
</tr>
<tr>
<td>AppartmentUnit</td>
<td>Building unit in ApartmentComplex that can be associated to person through right, being an object for registration.</td>
</tr>
<tr>
<td>Appurtenance</td>
<td>Something that is connected to a property in such way that it has to be transferred along it. Examples are e.g. diverse land use rights.</td>
</tr>
<tr>
<td>CodeList</td>
<td>States possible alternatives for an attribute and restricts in that way the values that can be assigned.</td>
</tr>
<tr>
<td>Conveyer</td>
<td>A person that is licensed to convey (notarise) legal documents.</td>
</tr>
<tr>
<td>Encumbrance</td>
<td>A claim, line charge, attached to and binding real property.</td>
</tr>
<tr>
<td>GroupPerson</td>
<td>Person that is an aggregation of many persons, each with own share. This is meant to encapsulate diverse nomadic rights in the CCDM.</td>
</tr>
<tr>
<td>LegalDocument</td>
<td>A document that is conveyanced and serves as a proof of evidence.</td>
</tr>
<tr>
<td>Members</td>
<td>Represents the association value between Person and GroupPerson.</td>
</tr>
<tr>
<td>MoneyProvider</td>
<td>Bank or other money lending institutions.</td>
</tr>
<tr>
<td>Mortgage</td>
<td>A legal instrument that creates a lien upon real estate securing the payment of a specific debt.</td>
</tr>
<tr>
<td>NaturalPerson</td>
<td>Human being</td>
</tr>
<tr>
<td>NonGeoRealEstate</td>
<td>Reserved for real estates that do not have fixed geometry like e.g. land-use rights (fishing, hunting) or when geometry is unknown.</td>
</tr>
<tr>
<td>NonNaturalPerson</td>
<td>Often referred as legal entity, examples are company or organisation.</td>
</tr>
<tr>
<td>NPPRegion</td>
<td>Stands for Non-Planar-Partitioning Region and is used parallel with PartitionParcel and ServingParcel to partition the complete planar domain of landownership. Within can NPPRegion parcels can be represented with spaghetti data or points.</td>
</tr>
<tr>
<td>Obligation</td>
<td>Duty or responsibility that comes with the property.</td>
</tr>
<tr>
<td>Parcel</td>
<td>Fundamental entity in cadastral systems. This class inherits both RealEstateObject and PartitionParcel, connecting the parcel registration with spatial delimitation.</td>
</tr>
<tr>
<td>ParcelBoundar</td>
<td>Between two parcels is at least one parcel boundary. It has the geometry GM_Curve as defined in ISO/TC211-19107 and 1:1 association with TP_Edge.</td>
</tr>
<tr>
<td>ParcelComplex</td>
<td>Aggregation of two or more parcels. Temporary intermediate level when merging parcels into new one.</td>
</tr>
<tr>
<td>PartitionParcel</td>
<td>Partitions the complete into no-overlapping parcels.</td>
</tr>
<tr>
<td>PartOfParcel</td>
<td>Composite of parcel. Represents the different parts after parcel-split. Temporary intermediate level when splitting a parcel into two/more.</td>
</tr>
<tr>
<td>Person</td>
<td>Is either natural or non-natural and has unique identifier.</td>
</tr>
<tr>
<td>PointParcel</td>
<td>Single coordinate pair to locate parcel when complete geometric extent is unknown. PointParcel is always within NPPRegion.</td>
</tr>
<tr>
<td>PublicRestOrAdv</td>
<td>Public decision can influence a property, either by giving the owner a restriction because of restrictive regulation, or an advantage e.g. like building permit.</td>
</tr>
<tr>
<td>RealEstateObject</td>
<td>RealEstateObject is the centre of the CCDM. It is an object subjected to cadastral system.</td>
</tr>
<tr>
<td>Classes</td>
<td>Definitions</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Registration</td>
<td>registration, bridging legal/administrative part of the CCDM with the geographic part.</td>
</tr>
<tr>
<td>Regulation</td>
<td>Restricts use of a property e.g. because of preservation (cultural or natural).</td>
</tr>
<tr>
<td>RestrictionArea</td>
<td>Is meant to register right and consequently restriction that can be to land use because of e.g. utilities or preservations.</td>
</tr>
<tr>
<td>RRR</td>
<td>Right, restrictions and responsibilities. This class is subjected to private laws opposite to the PublicRestOrAdv class that relies on public law.</td>
</tr>
<tr>
<td>ServingParcel</td>
<td>Is associated to at least two parcels and services them in a way by providing e.g. common playground or parking. Serving parcels belong not to persons but other parcels.</td>
</tr>
<tr>
<td>SourceDocument</td>
<td>This is a super class to LegalDocument and SurveyDocuments and assigns commonly important attributes to both classes.</td>
</tr>
<tr>
<td>SpaghettiParcel</td>
<td>Parcel that is made up with often inconsistent data not storing any topology. SpaghettiParcel is always within NPPRegion.</td>
</tr>
<tr>
<td>SurveyDocument</td>
<td>Survey points are published in a survey document that is made by a surveyor.</td>
</tr>
<tr>
<td>Surveyor</td>
<td>Person that is licensed to carry out a legal survey.</td>
</tr>
<tr>
<td>SurveyPoint</td>
<td>Point surveyed in the field. This class assigns diverse measurement parameters to the survey, like quality, projection etc.</td>
</tr>
<tr>
<td>TP_Edge</td>
<td>Topological edge as defined in ISO/TC 211-19107. Edge = “node-start, node-end, left-face, right-face, &lt;point&gt;”</td>
</tr>
<tr>
<td>TP_Face</td>
<td>Topological face as defined by ISO/TC211-19107. Face = &lt;edges&gt;</td>
</tr>
<tr>
<td>TP_Node</td>
<td>Topological node as defined by ISO/211-19107. Node = point</td>
</tr>
<tr>
<td>VolumeProperty</td>
<td>A thee-dimensional property that does not fit into the conventional planar registration of cadastres. Example of VolumeProperty is e.g. underground tunnel or a building property built on a bridge (does not have claim to the land beneath the bridge).</td>
</tr>
</tbody>
</table>
Appendix B: New version of the CCDM

The Legal/administrative and Person classes together

[181]
The different type of Immovable object classes together
Implementation of the Core Cadastral Domain Model in a Distributed Environment using OpenGIS Standards

The geometry, topology and some related packages, purples

[181]
Appendix C: Representation of the complete CCDM

The administrative/legal side of the CCDM as represented

[15]
The geographic side of the CCDM as represented in [15]
Appendix D: UML Model Dutch Cadastral Database-basic Classes

Source (TU Delft)