

**UMA EXTENSÃO SQL PARA O SUPORTE DE
TIPOS DE DADOS ESPACIAIS AVANÇADOS E
RESTRIÇÕES DE INTEGRIDADE**

LUÍS EDUARDO OLIVEIRA LIZARDO

**UMA EXTENSÃO SQL PARA O SUPORTE DE
TIPOS DE DADOS ESPACIAIS AVANÇADOS E
RESTRIÇÕES DE INTEGRIDADE**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: CLODOVEU AUGUSTO DAVIS JUNIOR

Belo Horizonte
Fevereiro de 2017

LUÍS EDUARDO OLIVEIRA LIZARDO

**A SQL EXTENSION TO SUPPORT ADVANCED
SPATIAL DATA TYPES AND INTEGRITY
CONSTRAINTS**

Dissertation presented to the Graduate Program in Computer Science of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

ADVISOR: CLODOVEU AUGUSTO DAVIS JUNIOR

Belo Horizonte

February 2017

© 2017, Luís Eduardo Oliveira Lizardo.
Todos os direitos reservados.

L789s Lizardo, Luís Eduardo Oliveira
A SQL extension to support advanced spatial data
types and integrity constraints / Luís Eduardo Oliveira
Lizardo. — Belo Horizonte, 2017
xxviii, 63 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de
Minas Gerais

Orientador: Clodoveu Augusto Davis Junior

1. Computação – Teses. 2. Sistemas de informação
geográfica. 3. Banco de dados espaciais. 4. Modelagem
de banco de dados. I. Orientador. II. Título.

CDU 519.6*72(043)



UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

A Sql Extension To Support Advanced Spatial Data Types And Integrity
Constraints

LUÍS EDUARDO OLIVEIRA LIZARDO

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. CLODOVEU AUGUSTO DAVIS JÚNIOR - Orientador
Departamento de Ciência da Computação - UFMG

PROF. GILBERTO RIBEIRO DE QUEIROZ
Divisão de Processamento de Imagens - INPE

DRA. KARLA ALBUQUERQUE DE VASCONCELOS BORGES
Prodabel

PROFA. MIRELLA MOURA MORO
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 22 de fevereiro de 2017.

In memory of my grandfathers:
Manuel de Oliveira Duarte and João Crisóstomo Lizardo.

Acknowledgments

I would like to thank my advisor and professor Clodoveu. Without his excellent guidance, patience and constant support, this work wouldn't have been completed.

For all the classes and technical learning, I am especially thankful for the professors and teachers, from EMNSA, CEFET-MG, FH-S and UFMG. Without your knowledge and patience I would not be finishing this important step in my life.

I would like to express my deep gratitude to my mother Alzira, who has inspired and motivated me all the times to achieve my goals. I owe you everything that I am today. I am extremely fortunate to be so blessed.

To my dear friends, I am very grateful to each of the tiny moments that we've spent together. The help and understanding of each one of you have been essential to my growth. Every moment with you is an eternal bond, and this is noticed in the human being each of us are and have become.

I extend my gratitude to all my research mates of the Lab CS+X and LBD. I am also thankful to Angélica, who helped me with the images of Sketch-Up.

“A aula nunca termina! / Learning never ends!”
(CEFET-MG 2006- ∞ Volleyball group)

Resumo

As primitivas geométricas definidas pelos padrões OGC e ISO, implementadas na maioria dos sistemas gerenciadores de banco de dados (SGBD) com suporte para dados espaciais, são incapazes de capturar a semântica de tipos ricos de representação, como os encontrados nos atuais modelos de dados geográficos. Além disso, os SGBDs relacionais não estendem os mecanismos de integridade referencial para cobrir relações espaciais e para suportar restrições de integridade espacial. Em vez disso, eles geralmente assumem que toda a verificação de integridade espacial será realizada pela aplicação, durante o processo de entrada de dados. Além de não ser prático, se o DBMS suportar muitas aplicações, isto pode levar a implementações redundantes e inconsistentes. Por isso, este trabalho apresenta o AST-PostGIS, uma extensão para PostgreSQL/PostGIS que incorpora à Linguagem SQL tipos de dados espaciais avançados e implementa restrições de integridade espaciais. A extensão reduz a distância entre os projetos conceituais e físicos de banco de dados espaciais, fornecendo representações ricas para geometrias do tipo geo-objeto e geo-campo. Ele também oferece funções para garantir a consistência das relações espaciais durante as atualizações de dados. Outras funções podem ainda ser utilizadas antes de impor restrições de integridade espacial pela primeira vez para verificar a consistência inicial do banco de dados. O uso do AST-PostGIS é ilustrado em um pequeno projeto de banco de dados geográfico urbano, mapeando seu esquema conceitual para a implementação física em SQL estendido.

Palavras-chave: OMT-G, Sistema de Informação Geográfica, Restrições de Integridade Espaciais, Banco de Dados Espaciais, Modelagem de Banco de Dados.

Abstract

Geometric primitives defined by OGC and ISO standards, implemented in most modern spatially-enabled database management systems (DBMS), are unable to capture the semantics of richer representation types, as found in current geographic data models. Moreover, relational DBMSs do not extend referential integrity mechanisms to cover spatial relationships and to support spatial integrity constraints. Rather, they usually assume that all spatial integrity checking will be carried out by the application, during the data entry process. This is not practical if the DBMS supports many applications, and can lead to redundant and inconsistent work. Therefore, this work presents AST-PostGIS, an extension for PostgreSQL/PostGIS that incorporates advanced spatial data types and implements spatial integrity constraints. The extension reduces the distance between the conceptual and the physical designs of spatial databases, by providing richer representations for geo-object and geo-field geometries. It also offers procedures to assert the consistency of spatial relationships during data updates. Such procedures can also be used before enforcing spatial integrity constraints for the first time. We illustrate the use of AST-PostGIS on an urban geographic database design problem, by mapping its conceptual schema to the physical implementation in extended SQL.

Keywords: OMT-G, Geographic Information Systems, Spatial Integrity Constraints, Spatial databases, Spatial databases modeling.

List of Figures

2.1	DE-9IM over spatial object interactions.	8
2.2	Conceptual diagram for a spatial RDBMS.	10
3.1	Simplified graphical notation of a class in OMT-G.	18
3.2	Geo-object classes	19
3.3	Geo-field classes	19
3.4	OMT-G notations for relationships.	21
3.5	OMT-G notations for generalizations and aggregations.	22
4.1	AST-PostGIS log table schema	35
5.1	OMT-G schema fragment for an urban geographic application	38
5.2	Overview of urban scenario without integrity constraints violations.	44
5.3	Spatial aggregation violation: a parcel extrapolates the block limits.	44
5.4	AST-PostGIS response for a spatial aggregation violation.	45
5.5	Spatial relationship violation: building is not within the parcel area.	45
5.6	AST-PostGIS response for a spatial relationship integrity constraint violation.	46
5.7	Arc-Node network violation: streets segments (arcs) do not intersect cross- ings (nodes)	46
5.8	AST-PostGIS response for a street network integrity constraint violation.	46

List of Tables

3.1	Geometric types: OMT-G and OGC SFS	27
4.1	Advanced spatial data types supported by AST-PostGIS	30
4.2	Consistency check functions supported by AST-PostGIS	34
5.1	Spatial integrity constraints derived from the schema in Figure 5.1	40

List of Listings

4.1	Aggregation Trigger	32
4.2	Arc-Node Trigger	32
4.3	Spatial Relationship Trigger	33
4.4	Usage of the consistency check function for network relationship	34
5.1	SQL schema for the tables definition	41
5.2	Integrity constraints implementation for aggregations	41
5.3	Integrity constraints implementation for spatial relationships	42
5.4	Integrity constraint implementation for arc-node network	43
A.1	Tables definition	57
A.2	Indexes definition	60
A.3	Triggers definition	62

Acronyms

ACID: Atomicity, Consistency, Isolation, Durability

AST: Advanced Spatial Type

DBMS: Database Management System

DDL: Data Definition Language

DE-9IM: Dimensionally-Extended Nine-Intersection Model

ER: Entity–Relationship

GIS: Geographic information system

GML: Geography Markup Language

ISO: International Organization for Standardization

MVCC: Multi-Version Concurrency Control

OGC: Open Geospatial Consortium

OMT-G: Object Modeling Technique for Geographic Applications

OMT: Object Modeling Technique

PL/pgSQL: Procedural Language/PostgreSQL

RDBMS: Relational Database Management System

SFS4SQL: Simple Features Specification for SQL

SQL: Structured Query Language

SRID: Spatial Reference System Identifier

TCL: Tool Command Language

TIN: Triangular Irregular Network

UML: Unified Modeling Language

XML: Extensible Markup Language

Contents

Acknowledgments	xi
Resumo	xv
Abstract	xvii
List of Figures	xix
List of Tables	xxi
List of Listings	xxiii
Acronyms	xxv
1 Introduction	1
1.1 Contributions and objective	3
1.2 Work outline	4
2 Related Work	5
2.1 Spatial data	5
2.2 Spatial relationships	7
2.3 Spatial integrity constraints	8
2.4 Spatial database management systems	9
2.4.1 PostgreSQL Overview	11
2.5 Conceptual modeling for spatial databases	13
3 The OMT-G Model	17
3.1 Class diagram	18
3.1.1 Class structure	18
3.1.2 Relationships	19

3.2	Integrity Constraints	21
3.2.1	Geo-field constraints	22
3.2.2	Geo-object constraints	23
3.2.3	Network relationship constraints	24
3.2.4	Spatial aggregation constraint	24
3.2.5	Spatial relationship constraints	25
3.3	Mapping OMT-G to object-relational spatial databases	27
3.4	Further work on OMT-G	28
4	AST-PostGIS	29
4.1	Advanced Spatial Data Types	30
4.2	Integrity constraints for spatial relationships	30
4.3	Consistency Check Functions	34
4.4	Summary of limitations	35
5	Case Study: Urban geographic database	37
5.1	Conceptual schema	37
5.2	Physical implementation	39
5.3	AST-PostGIS responses	43
5.3.1	Spatial aggregation violation	44
5.3.2	Spatial relationship violation	45
5.3.3	Arc-node network violation	45
6	Conclusions and Future Work	47
	Bibliography	49
	Appendix A SQL schema for the urban cadastral database	57

Chapter 1

Introduction

OpenGIS (OGC) and SQL/MM (ISO) standards have been instrumental in the effort to standardize spatial data management using relational and object-relational databases. OpenGIS, the set of spatial data standards proposed by the Open Geospatial Consortium (OGC), covers many aspects of spatial data representation, spatial databases and Web services. The Simple Features Specification for SQL standard (SFS4SQL) [OGC 06-104r4, 2010; OGC 06-103r4, 2011], a component of OpenGIS, defines a standard SQL schema for storing, retrieving, querying and updating geospatial features in relational database management systems (DBMSs). Using SFS4SQL, geospatial objects are represented by a geometric shape, which in turn uses a spatial reference system for geographic coordinates. SFS4SQL supports a limited number of basic geometric representations, such as points, linestrings and polygons, introduces multipoints, multilinestrings and multipolygons, and also allows heterogeneous geometry collections. The standard specifies some forms of geometric constraints, such as the detection of simple and non-simple linestrings, and establishes the Dimensionally-Extended Nine-Intersection Model (DE-9IM) [Clementini et al., 1993] as the basis for detecting and enforcing topological relationships.

The SQL/MM standard (Part Three - Spatial) [ISO/IEC 13249-3, 2016; Melton and Eisenberg, 2001] is derived from OpenGIS and provides more functions and enables more dimensions for objects. Its definitions include considerations on how spatial data are to be represented as values and which functions must be used to compare, transform and process spatial data in various ways [Stolze, 2003].

Since 1999, these standards have been progressively adopted by popular DBMSs. For instance, Oracle Spatial¹ complies with OpenGIS SFS and supports SQL/MM types

¹<https://www.oracle.com/database/spatial/>

and operators [Oracle, 2017]. MySQL Spatial² extension implements only part of the OpenGIS standard and only the 2D representations without reference sets [Piórkowski, 2011; Widenius and Axmark, 2002]. Microsoft’s SQL Server Spatial Storage³ conforms its geometry data types to the OpenGIS SFS, but not its geography data types [Aitchison, 2009; Fang et al., 2008]. IBM DB2 Spatial Extender⁴ implements types and functions defined by both specifications [Adler, 2001]. PostGIS⁵, the open source spatial extension for PostgreSQL⁶, conforms to both standards almost completely [Obe and Hsu, 2015; Piórkowski, 2011].

Adopting standards has been hugely beneficial for the spread of geospatial data in information systems, as they promote interoperability among spatial DBMSs. On the other hand, the representations defined by those standards are restricted to geometric primitives, devoid of more complex geographic behavior. Take, for instance, the representation of geographic networks, such as a water distribution system or a transportation network. While it is possible for system designers to use OGC *points* to represent nodes and *linestrings* to represent arcs, the role of such *points* and *linestrings* in the database should also include network connectivity. However, there is no way to make such role explicit in SFS-extended SQL, with the statements that are used to define the database’s structure and expected behavior, except by coding custom spatial integrity constraints. If network nodes and arcs were defined as primitive types, it would be possible to directly create geographic network structures in the physical phase of database design. As long as they are represented with the corresponding spatial integrity constraints, thus defining geospatial behavior on top of the geometric representation. This strongly contrasts with, for instance, the mechanisms that allow specifying referential integrity constraints in conventional SQL.

Another example is the mapping of planar subdivisions, sets of polygons in which (1) no polygons overlap, (2) no gaps between polygons exist, and (3) the union of the polygons covers the entire geographic area of interest for the application. Planar subdivisions are common in the conceptual design of geographic applications. They can be used to represent territorial hierarchies of various types and many kinds of environmental classifications, such as vegetation or soil type. Clearly, the simple mapping of a planar subdivision class, as specified in conceptual design, into a table containing polygon geometries is insufficient to fulfill the designer’s intentions and needs. Spatial integrity constraints would have to be enforced by the DMBS, so the semantics of

²<https://dev.mysql.com>

³<https://www.microsoft.com/sql-server>

⁴<http://www-03.ibm.com/software/products/en/db2spaext>

⁵<http://www.postgis.net/>

⁶<https://www.postgresql.org/>

planar subdivisions is adequately implemented. If a planar subdivision representation was available in SFS4SQL, constraints (1-3) could be previously implemented, and operationally enforced as any other integrity constraint in the database.

In contrast, conceptual geographic data models, such as OMT-G [Borges et al., 2001], GeoOOA [Kosters et al., 1997], MODUL-R [Bédard et al., 1996] etc, offer semantically rich spatial object classes and relationships. For instance, OMT-G provides primitives for modeling planar subdivisions, triangular irregular networks, samples, isolines, and tessellations, supporting spatial and conceptual generalizations, topological relationships, “whole-part” structures, networks and multiple representations of objects. There are no directly corresponding physical representation alternatives for such primitives in current spatial database management systems.

We argue, therefore, that the gap that separates conceptual design models and physical implementation data types imposes a more thorough mapping process, in which spatial integrity constraints can be extracted and detailed, so the semantics of conceptual design classes can be adequately implemented in the DBMS. From observation and personal practice, we notice that most spatial integrity constraints can be generalized and implemented using SQL tools such as checks, triggers and assertions, with the help of SFS standard functions. Implementing and re-implementing such generic constraint verification code, on the other hand, is tedious and error-prone. Furthermore, it is difficult to perceive a connection to the conceptual schema by reading the SQL data definition language (DDL) code that specifies the database’s structure.

1.1 Contributions and objective

In this work we propose AST-PostGIS, a SQL extension that implements advanced spatial data types and spatial integrity constraints in a RDBMS capable of storing and processing spatial data. Our objective is to reduce the distance between the conceptual and physical schemas of spatial database design. Our extension also provides mechanisms to identify integrity constraint violations on spatial databases upon initial enforcement of constraints, and trigger procedures that constrain the insertion or deletion of inconsistent spatial data in the RDBMS, following conceptual design semantics.

The AST-PostGIS extension is open-source⁷ and it is currently available for PostgreSQL, an object-relational database system [Momjian, 2001], to expand its spatial extension PostGIS. The extension is cross-platform and easy to install and enable in each database schema. The implemented spatial data types and integrity constraints

⁷https://github.com/lab-csx-ufmg/ast_postgis

are based on those defined on OMT-G, an object oriented data model for the design of geographic applications and geographic database systems [Borges et al., 2001].

1.2 Work outline

The remainder of this work is organized as follows.

Chapter 2 – Related Work: Covers literature that is relevant to our proposal and explores some basic concepts.

Chapter 3 – The OMT-G Model: Presents the OMT-G data model and gives a brief overview of its primitives for conceptual modeling of geographic data.

Chapter 4 – AST-PostGIS: Introduces the AST-PosGIS, our SQL extension that implements advanced spatial data types and integrity constraints for PostgreSQL/PostGIS.

Chapter 5 – Case Study: Urban geographic database: Illustrates the use of the AST-PostGIS with a implementation of a small urban geographic database schema.

Chapter 6 – Conclusions and Future Work: Discusses the conclusions and future work.

Chapter 2

Related Work

In this chapter, we review relevant concepts and discuss topics related to this work. We start with Section 2.1 by explaining the characteristics of spatial data. Then, we discuss spatial relationships and explain the methods proposed to enumerate all existent relationships in Section 2.2. Next, we show the classification of spatial integrity constraints in Section 2.3, which are the main topic of our work. In Section 2.4, we discuss the peculiarities of spatial database management systems and give a brief overview of PostgreSQL and PostGIS, the RDBMS for which we developed AST-PostGIS. Finally, in Section 2.5, we review the conceptual data models existent for spatial databases.

2.1 Spatial data

Spatial data describe phenomena to which some spatial dimension is associated. *Geographic* or *georeferenced* data are spatial data in which the spatial dimension is associated with their location on the earth's surface, at a given time or period of time [Câmara et al., 1996; Borges, 1997]. Spatial data consist of spatial objects made up of points, lines, regions, rectangles, surfaces, volumes, and even data of higher dimensionality, which can include time (spatiotemporal data). Examples of spatial data include representations of cities, rivers, roads, countries, states, sole coverage, mountain ranges etc. Examples of spatial properties include the length of a given river, or the surface area of a given country. Often it is also desirable to attach non-spatial attribute information, such as elevation heights, city names etc., to the spatial data. The spatial representation of a geographical entity is the description of its geometric shape, associated with the geographical position.

Geographic data have *geometric* and *topological* properties. Metric relationships can be established from the primitives of geometric features. These relationships ex-

press metric features, such as distances, with reference to a coordinate system. Based on the geometry, some geometric properties, such as length, sinuosity and line orientation can be established. For instance, perimeter and surface area for polygons, volume for three-dimensional entities, and shape and slope for both lines and polygons. Topological properties (non-metric) are based in relative positions of objects in the space, and include connectivity, orientation (from, to), adjacency and contention. Some geographic entities have topological properties that are invariant by elastic deformations. For example, the connectivity between road intersections and road segments is kept independently of the coordinate system or projection space used. Some spatial concepts can even be measured in both domains: topological and geometric. *proximity*, for example, can be obtained both through adjacency and Euclidean distance [Laurini and Thompson, 1992].

Vector and *raster* data are the two primary geometric types. Vector data are comprised of vertices and paths. The three basic types used for vector data are *points*, *lines* and *polygons*, all of which are representations of the space occupied by real-world entities. Points are simply pairs of *XY* coordinates (such as latitude and longitude). When each point is connected with a line in a particular order, they become a vector line feature. Lines usually represent features that are linear in nature, such as rivers and roads. When a set of points are joined in a particular order and closed, they become a vector polygon feature. In order to create a polygon, the first and last coordinate pairs coincide and all other pairs must be unique. Polygons represent features that cover a two-dimensional area. Examples of polygons are buildings, agricultural fields and discrete administrative areas.

Raster data is made up of pixels (also referred to as grid cells). They are usually regularly-spaced and square. Each pixel is associated with a value or class. Raster models are useful for storing data that are obtained using regularly spaced samples, as in an aerial photography, an elevation surface or a satellite image. Raster data models can be discrete and continuous. Discrete rasters are also referred to as thematic or categorical raster data. They have distinct themes or categories. For example, one grid cell represents a land cover class or a soil type. Each class can be discretely defined as to where it begins and ends. Discrete data usually consist of integers to represent classes. Continuous rasters, on the other hand, are grid cells with gradual changing data such as elevation, temperature or an aerial photograph. Continuous data is also known as non-discrete or surface data. A continuous raster surface can be derived from a fixed registration point. For example, a digital elevation model is measured from sea level. Each cell represents a value above or below sea level. An aspect cell value is derived from a fixed direction such as north, east, south or west.

2.2 Spatial relationships

Pullar and Egenhofer [1988] classified spatial relationships in several classes: *direction* relationships describe order in space (e.g. *north_of*, *northeast_of*), *topological* relationships describe neighborhood and incidence (e.g. *adjacent*, *inside*, and *disjoint*), *comparative* or *ordinal* relationships describe inclusion or preference (e.g. *in*, *at*), *distance* relationships, such as *far* and *near*, and *fuzzy* relationships, such as *next to* and *close to*, describe relative metric position.

Among these, topological relationships are the most fundamental and have been studied in more depth [Güting, 1994]. A basic question is whether it is possible to enumerate all existent relationships. Egenhofer [1989] and Egenhofer and Herring [1990] proposed simple method for this, considering the intersections between two polygons as to their interiors and boundaries, configuring a *4-intersection matrix*. Egenhofer and Franzosa [1991] showed that for two objects there are four intersection sets; each of them may be empty or non-empty, which leads to $2^4 = 16$ combinations. Eight of them are not valid, and two of these are symmetric, so that six different relationships result, named *disjoint*, *in*, *touch*, *equal*, *cover* and *overlap*.

This approach has been extended to support other types of geometries. For example, point and line features have been added [Egenhofer and Herring, 1994; Hoop and Van Oosterom, 1992]. Clementini et al. [1993] also considered the spatial dimension of the intersection (called the dimension-extended method). This model has been used as the standard to compare two geospatial objects. It translates the relationship between two geometries into a set of outcomes based on a decision tree. An example of the result matrix, called the Dimensionally Extended Nine-Intersection Model (DE-9IM) [Strobl, 2008], can be seen in Figure 2.1.

The intersection of the interiors is a two-dimensional area, so that matrix cell's value equals 2, showing that the object resulting from that operation is a polygon, i.e., a two-dimensional object. When intersections are over single lines, that matrix cell's value equals 1, indicating that a one dimensional object is the result. When the polygons touch over single points, that portion of the matrix equals 0, which indicates the zero-dimensional point object as result. When there is no intersection between components, the respective matrix value is set to a boolean false. Likewise, when any kind of intersection is sufficient to configure a relationship, a boolean true is used.

The DE-9IM model has been adopted by the OGC [OGC 06-103r4, 2011] and implemented in OGC-compliant spatial database management systems, such as PostGIS [Obe and Hsu, 2015].

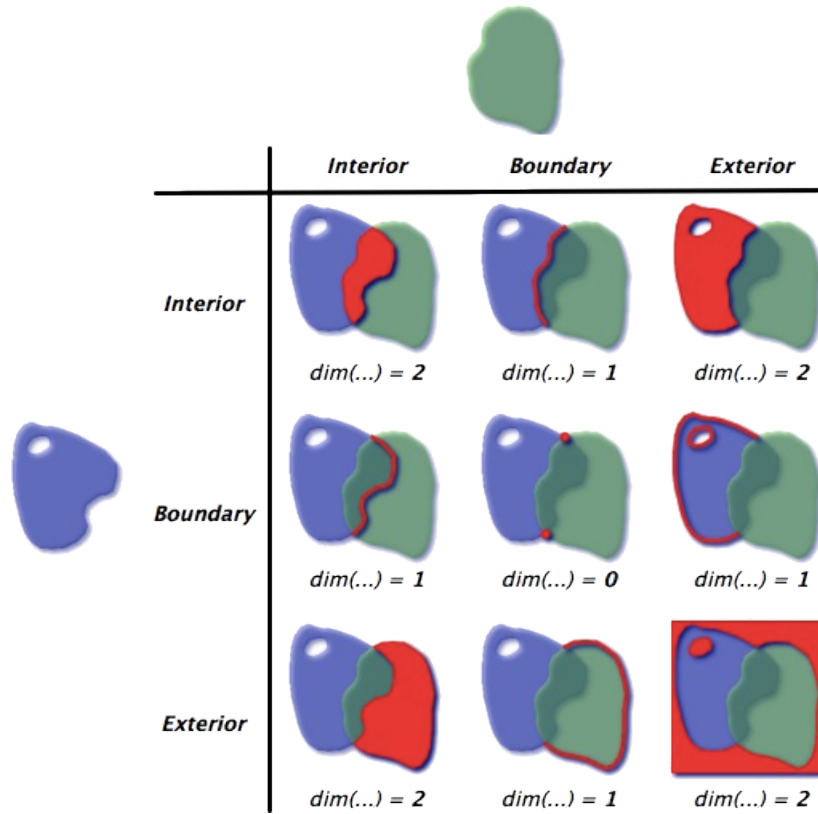


Figure 2.1: DE-9IM over spatial object interactions.

Source: Boundless Suite User Manual [2016]

2.3 Spatial integrity constraints

Integrity constraints are fundamentally important for database management. They specify those configurations of the data that are considered semantically correct, a property that databases are required to satisfy at any time in order to be consistent. The simplest example of this is specifying that a data item must be of a certain type, restricted to a limited value or identified by a unique key. More complex integrity constraints can also be specified to manage, for example, the relationships between database records. Such examples of integrity constraints – namely *domain*, *value*, *key* and *relationship* – are well supported by modern RDBMSs and are extensively documented [Melton and Simon, 1993; Elmasri and Navathe, 2015].

In the scope of spatial databases, integrity constraints are also related to the topological and geometrical aspects of the data [Borges et al., 2002, 2005]. Cockcroft [1997] proposed a classification to cover the peculiarities of spatial data and defined the spatial integrity constraints as *topological*, *semantic* or *user rules*, as follows:

Topological integrity constraints are related to the geometrical properties and spatial relations [Egenhofer and Franzosa, 1991]. These principles are applied to application-specific entities and relationships to provide the basis for integrity control. Modeling city neighborhoods is an example of this constraint: one neighborhood must be contained within the city limits, and there must be no point in the municipal area that does not belong to a neighborhood. Likewise, a neighborhood cannot belong to multiple cities.

Semantic integrity constraints are concerned with the meaning of geographic features. These constraints verify if a database state is valid due to the properties of the objects that need to be stored. As an example is a rule in which a building cannot be intercepted by a street segment.

User-defined integrity constraints allow database consistency to be maintained as defined by the equivalent of “business rules” in non-spatial DBMS. The location of a gas station, which must stay 300 meters farther from any existing school for safety reason, is an example of this type of constraint. The municipal permitting process must consider this limitation in its analysis. User-defined rules may be stored and enforced by an active repository.

2.4 Spatial database management systems

A spatial DBMS is adapted to store and query geographic information. Most spatial DBMSs allow representing simple geometric objects such as points, lines and polygons. Some of them are able to handle more complex structures, such as 3D objects, topological coverages, linear networks, and triangular irregular networks. While conventional relational databases have developed to manage various numeric and character types of data, such databases require additional functionality to process spatial data types efficiently [Surve and Kathane, 2014].

The logic in a spatial DBMS normally consists of enabling infrastructure in the form of spatial indices maintained by the DBMS together with a collection of spatial operators [Güting, 1994]. A spatial index is a specialized form of relational DBMS index maintained for the geometry type, so that it is possible to quickly retrieve objects based upon their spatial characteristics, such as the location and extent of a particular object [Güting, 1994]. Spatial operators are normally functions that ascertain spatial relationships, such as finding all objects that are contained within a given object [Clementini and Di Felice, 2000]. Figure 2.2 schematically illustrates the additional functions and modules in a spatial RDBMS.

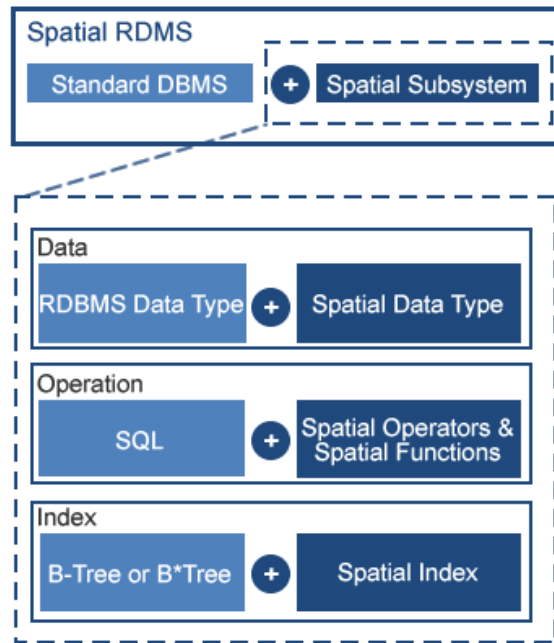


Figure 2.2: Conceptual diagram for a spatial RDBMS.

Source: [CUBRID Blog, 2016]

Many benefits follow if an existing relational DBMS is extended to process spatial data. First, even when conducting geospatial tasks, there will be many occasions when conventional data types, such as numbers or characters, are used with no modification in relation to non-spatial DBMSs. Another benefit is that there will not be a burden of additional training, since SQL is a traditional and very well-known solution to manage and query the data.

Relational DBMS is not the only type of database management system available [Han et al., 2011]. Likewise, spatial RDBMS is not the only type of spatial database management system available [Queiroz et al., 2013]. Many DBMSs, such as MongoDB¹, a document-oriented database, and search engines such as Lucene² or Solr³, provide spatial data processing features. However, these solutions offer less features and do not provide high precision calculations [Lizardo et al., 2014; Santos et al., 2015]. Currently, spatially-extended relational DBMSs, such as PostGreSQL/PostGIS and Oracle Spatial, are more broadly used than any other available.

¹MongoDB: <https://www.mongodb.com/>

²Apache Lucene: <http://lucene.apache.org/>

³Apache Solr: <http://lucene.apache.org/solr/>

2.4.1 PostgreSQL Overview

PostgreSQL is a powerful and one of the most successful open source object-relational databases available. It is arguably also the most advanced, with a wide range of features that challenge even many closed-source databases [Souza et al., 2011; Douglas and Douglas, 2005; Momjian, 2001]. Here we cite few of the features found in a standard PostgreSQL distribution:

Object-relational. In PostgreSQL, every table defines a class. It implements inheritance between tables. Functions and operators are polymorphic.

Standards compliant. PostgreSQL syntax implements most of the SQL92 standard and many features of SQL99. Where differences in syntax occur, they are most often related to features unique to PostgreSQL.

Open source. An international team of developers maintains PostgreSQL. Its core team has been working on enhancing PostgreSQL's performance and feature set since at least 1996.

Transaction processing. PostgreSQL protects data and coordinates from multiple concurrent users through full transaction processing. The transaction model used by PostgreSQL is based on multi-version concurrency control (MVCC).

Referential integrity. PostgreSQL is fully ACID compliant and implements complete referential integrity by supporting foreign and primary key relationships, joins, views, triggers and stored procedures. Business rules can be expressed *within* the database rather than relying on external tools.

Multiple procedural languages. Triggers and other procedures can be written in PL/pgSQL, a procedural language similar to Oracle's PL/SQL⁴. But it is also possible to develop server-side code in Tcl, Perl and even Bash.

Multiple-client APIs. PostgreSQL supports the development of client applications in C/C++, Java, .Net, Perl, Python, Ruby, Tcl, ODBC, among others, with excellent documentation.

Unique data types. PostgreSQL provides a variety of data types, besides the usual numeric, string, and date types, it provides geometric types, boolean data types, and data types designed specifically to deal with network addresses. It also supports storage of binary large objects, including pictures, sounds and video.

⁴Oracle PL/SQL: <http://www.oracle.com/technetwork/database/features/plsql/index.html>

Extensibility. One of the most important features of PostgreSQL is that it can be extended. It is possible to write custom data types, new functions and operators, and even new procedural and client languages. Many contributed packages are available on the Internet⁵. In this work, we use the extensibility features of PostgreSQL to develop AST-PostGIS.

2.4.1.1 PostGIS: the spatial database extension for PostgreSQL

PostGIS is a spatial database extension for PostgreSQL [Ramsey et al., 2005]. PostGIS “spatially enables” the PostgreSQL server, allowing it to be used as a back-end spatial database for geographic information systems (GIS) and web-mapping applications in the same manner as Microsoft’s SQL Server Spatial and Oracle’s Spatial database extension. PostGIS follows the OGC/SQL-MM standards [Obe and Hsu, 2015]. Among its many features, we can cite:

High performance. PostGIS uses the smallest possible representations of geometry and index structures to maximize performance. PostGIS users have compared performance with proprietary databases on massive spatial data sets and PostGIS comes out on top [Zhou et al., 2009; Shukla et al., 2016]. Smaller data representations reduce throughput to slow hard-disks and keep more data in fast memory cache, then improving speed directly.

Spatial query. PostGIS includes a full set of geometry query operations: distance, containment, intersection, and full topological relationship matrices. In addition, queries are speed up by a self-tuning R-Tree index, fully integrated into the PostgreSQL’s query planner.

Data integrity. Storing spatial data in a database allows simple random access via any tool with SQL: scripts, desktop applications, other databases, Web services. PostGIS uses the PostgreSQL’s row-level locking to allow multiple processes to write in the spatial tables without resource contention and with guaranteed data integrity.

Spatial Analysis. Advanced GIS analysis can be carried out, using spatial joins, buffers, intersections, polygon building, line building, linear referencing and more.

⁵PGXN: PostgreSQL Extension Network: <http://pgxn.org/>

2.5 Conceptual modeling for spatial databases

The first data models for spatial applications were guided by existing GIS internal structures. The user interpretations of spatial phenomena were then forcibly adjusted to any structures available. The modeling process did not offer mechanisms that would allow for the representation of the reality according to the user's mental model [Borges et al., 2005]. Conventional semantic and object-oriented data models, like ER [Chen, 1976], OMT [Rumbaugh et al., 1991], IFO [Abiteboul and Hull, 1987] and UML [Booch et al., 2005] have also been used for modeling spatial databases. Although these models are highly expressive, they have limitations for modeling spatial applications, since they do not present geographic primitives for a satisfactory representation of spatial data and its peculiar properties.

Using such traditional models brought difficulties, because many spatial applications need to deal with specific aspects of spatial data, such as location, geometric constraints, time of observation and accuracy [Oliveira et al., 1997]. In conventional models it is not possible to differentiate between object classes that have a geographic reference and normal alphanumeric classes. Furthermore, it is difficult to represent the spatial relationships that exist as a consequence of the geometric nature of objects. Spatial relations are abstractions that help understand how objects relate to each other in the real world [Frank and Mark, 1991] and they need to be explicitly represented in the application's schema in order to make the schema easier to understand.

Therefore, modeling spatial databases is perceived to be more complex than modeling conventional databases, due to particular characteristics of geographic data. Modeling spatial data requires specific models that are capable of catching the semantics of geographic data, offering mechanisms of higher abstraction and implementation independence. Concepts like geometry and topology are fundamental in determining spatial relationships between objects. Moreover, spatial data have diverse origins and environmental data are a good example of such diversity. Elevation and soil properties, for example, vary continuously in space, while geological faults and river networks can be discretized. Depending on the level of detail considered, some real-world entities can even belong to both categories [Kemp, 1992].

Various spatial data models have been proposed in the literature. For example, Worboys et al. [1990] proposed EXT.IFO, a IFO Model [Abiteboul and Hull, 1987] extension with basic spatial types: point, line and polygon. However, fields, spatial aggregations, multiple views and other fundamental geographic modeling constructs are not represented in the model. Abrantes and Carapuça [1994] extended the OMT Data Model [Booch et al., 2005] and created OMT EXT, with primitives for modeling

topological relationships, namely *partition*, *covering* and *disconnected class*. The later being a concept associated with the subclasses derived from *partitions* and *coverings*.

GISER [Shekhar et al., 1997] and GMOD [Oliveira et al., 1997] do not define specific modeling primitives and cannot be considered proper data models [Borges et al., 2001]. They only provide standards to be followed by geographic application designers. GISER extends the ER Model [Chen, 1976] and integrates field-based and object-based models of geographic data by using the *discretized-by* relationship between feature fields and coverage entities. In addition, it has predefined entities and relationships that represent fields, objects view, network relationships and multiple visualization forms of an entity. GMOD allows, through predefined classes, the definition of georeferenced phenomena according to fields and objects. It also has predefined classes to model the geometry of spatial entities, as well as temporal dimension. GMOD introduces new relationships between entities, e.g. (*causal* and *version*). Both, GISER and GMOD are difficult to represent simultaneous conditions for a given entity due to the lack of specific primitives.

Bédard et al. [1996] introduced MODUL-R with a fixed set of geometric types that use spatial pictograms to represent the geometric shapes of entities. The combination of these pictograms can represent multiple views of the same entity. This model, on the other hand, does not distinguish between fields and objects and does not have primitives to represent topological connectivity, neither spatial aggregation. Another geographic data model, GeoOOA [Kosters et al., 1997], supports spatial and temporal class types, topological “whole-part” structures, network structures and a set of geometric types with the use of pictograms. Object classes with or without spatial representation are distinguished by this model. However, GeoOOA lacks the support for spatial integrity constraints and does not represent properly fields neither multiple ways to visualize an object.

Lisboa Filho and Iochpe [1999] proposed GeoFrame, a conceptual geographical model with a hierarchical class structure. The hierarchy is subdivided in the levels: *Planning*, *Metamodel* and *Spatial Representation*. In the planning level, the basic class is the *GeographicRegion*, which defines the regions of interest. For each region, it is defined associated themes (*Theme* class), such as limits of the urban area, hydrograph, public transport, road network, relief etc. A theme can also be subdivided in a hierarchy of sub-themes. The metamodel level is composed of meta-classes that reflect how the reality is interpreted and it can be represented by conventional data (*Conventional Object*) or geographic phenomena (*Geographic Phenomenon*). The latter being specialized in meta-classes for field (*GeographicField*) and objects (*GeographicObject*) views. The level of spatial representation reflects how the reality is represented by

designers and users in relation to the representation within the database. The *SpatialObject* class generalizes spatial representation classes observed in objects view (e.g. *Point*, *Line*, *Polygon* and *ComplexSpatialObj*), and the field view classes (e.g. *GridOfCells*, *AdjPolygons*, *Isolines*, *GridOfPoints*, *TIN* and *IrregularPoints*) are generalized by the *FieldRepresentation* class. Following works introduced tools to support the model [Lisboa Filho et al., 1999, 2004a,b].

To the best of our knowledge, the most complete spatial data model is OMT-G [Borges et al., 2001], because it is the only one that includes class, transformation and presentation diagrams for the modeling of geographic applications. It also supports topological, semantic and user-defined integrity constraints, along with primitives for the representation of multiple views. OMT-G differentiates between spatial relationships and simple associations and OMT-G diagrams tend to be smaller than others, because of the higher semantic content of its primitives. OMT-G model is described in more detail in Chapter 3 of this work.

Chapter 3

The OMT-G Model

OMT-G (*Object Modeling Technique for Geographic Applications*) [Borges et al., 2001] is an object-oriented data model for the design of geographic applications and geographic database systems. Derived from the UML [Booch et al., 2005] class diagram, OMT-G introduces primitives for modeling the geometric shape and location of geographic objects, supporting spatial and topological relationships, “whole-part” structures, networks, and multiple representations of objects and spatial relationships. Besides, the model allows the specification of alphanumeric attributes and associated methods for each class. The OMT-G data model reduces the distance between the conceptual project and the physical implementation of geographic applications, by allowing a more precise definition of the required objects, operations and visualization parameters.

Three main concepts sustain the OMT-G model: *classes*, *relationships*, and *spatial integrity constraints*. Classes and relationships define the basic primitives that are used to create application static schemas. The spatial integrity constraints ensure the necessary conditions to keep the database always consistent. OMT-G proposes three different diagrams for the modeling of a geographic application. The *class diagram*, which is the most used and specifies all classes along with their representations and relationships. From this diagram, a set of spatial integrity constraints that must be observed in the implementation can be derived. The *transformation diagram* is built when the application involves the derivation of some class from others or when the class diagram indicates the need for multiple representations of any class. With it, all transformation process can be specified, allowing the identification of any required methods for the implementation. Finally, a *presentation diagram* is built to provide guidelines for the visual aspect of objects in the implementation. Several visual aspects can be there for any given class, which allows for the definition of a view or set of views

for each application or group of users.

In the following sections we discuss the class diagram primitives and the spatial integrity constraints that can be derived from these primitives. Transformation and presentation diagrams, however, are not further discussed in this work as they are not related to the AST-PostGIS. More details about them is found in Borges et al. [2001]. In Section 3.3, we explain how OMT-G is mapped to object-relational spatial databases and discuss the loss of semantics imposed by the mapping process. Finally, in Section 3.4 we cite other works from our research group intended to foster the further use of the OMT-G data model.

3.1 Class diagram

Class diagrams are used in OMT-G to describe the structure and contents of a geographic database. A class diagram contains fixed rules and descriptions that define, for the conceptual representation, how the data are to be structured, including information on the representation that is to be adopted for each class.

3.1.1 Class structure

OMT-G specifies two types of classes in the class diagram: *georeferenced* and *conventional*. The georeferenced class notation have a top left-hand rectangle that points the geometry of the representation, whereas the notation used for conventional classes is similar to the notation used in the UML [Booch et al., 2005], as it is shown in Figure 3.1.

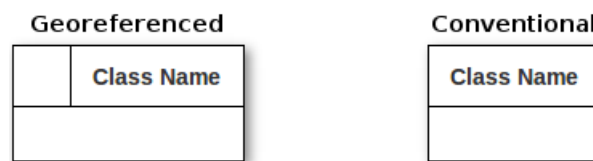


Figure 3.1: Simplified graphical notation of a class in OMT-G.

The distinction between conventional and georeferenced classes allows different applications to share spatial and non-spatial data, thus making it easier to develop integrated applications. Conventional classes have no geographical properties. Georeferenced classes, otherwise, include a geographical representation alternative, which specializes in two types of representations: discrete, associated with real world elements (*geo-objects*), or continuously distributed over the space (*geo-fields*). The geo-objects are represented by points, lines, polygons or network elements (nodes, unidirectional

and bidirectional arcs). Geo-fields correspond to variables such as soil type, relief and temperature, often seen as a surface, and can be represented by isolines, tessellation, planar subdivision, sampling or triangular irregular network (TIN). Figures 3.2 and 3.3 show, respectively, examples of geo-object and geo-field classes.

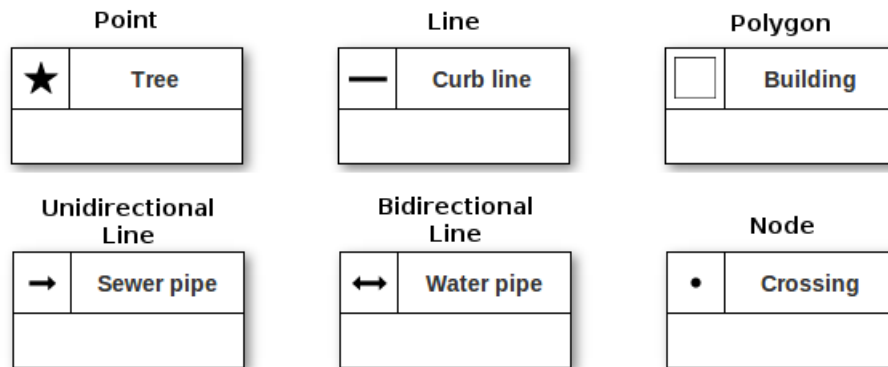


Figure 3.2: Geo-object classes

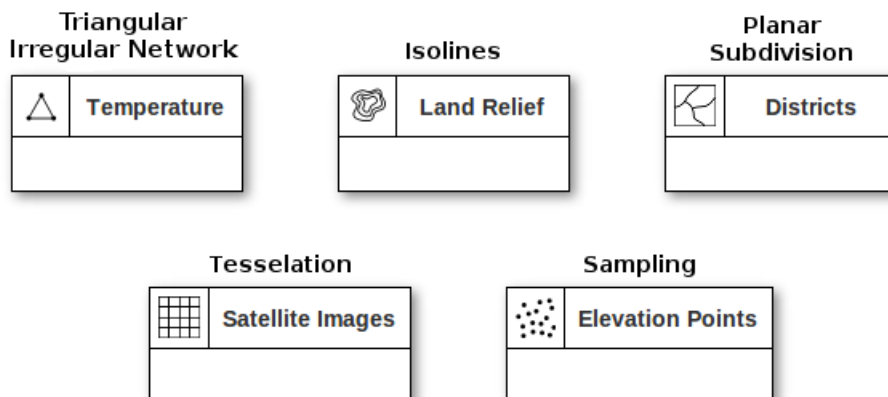


Figure 3.3: Geo-field classes

3.1.2 Relationships

OMT-G represents three basic types of relationships that can occur between its classes: *simple associations*, *network relationships* and *spatial relationships*, along with object-oriented relationships (*generalization/specialization*, *aggregation* and *conceptual generalization*).

Simple associations represent structural conventional relationships between objects of different classes, as in UML. *Network relations* are relationships among con-

nected objects. This type of relationship only shows the need for a logical connection, not a requirement for the implementation of a particular structure. A sewage arc-node network is an example of this type of relation. The arcs represent the piping segments while the nodes are used to represent network elements such as manhole, sewage treatment station and discharge. *Spatial relations* represent the topological, metric, ordinal and fuzzy relationships. Some relationships, like topological, can be derived automatically from the geometry of each object during the execution of data entry or spatial analysis operations. Others, called explicit relationships, need to be specified by the user in order to allow the system to store and maintain that information. OMT-G considers a set of nine different spatial relationships between georeferenced classes: *touch*, *in*, *cross*, *overlap*, *disjoint*, *adjacent to*, *coincide*, *contain* and *near*.

OMT-G makes it simple to distinguish between simple associations, spatial and network relationships. Simple associations are represented by continuous lines, whereas spatial relationships are indicated by dashed lines. Both relationships are characterized by their cardinality. The notation for cardinality adopted by OMT-G is the same used by UML. The network relationships are indicated by two parallel dashed lines, linking a node class to an arc class. Network structures without nodes can be indicated by a recursive relationship on the class which represents graph segments. The name given to the network is annotated between the two dashed lines. Figure 3.4 shows the OMT-G notations for relationships.

Generalization and *specialization* relationships present similar behavior to conventional object-oriented hierarchies. They apply to both georeferenced and conventional classes, following the definitions and notation proposed for UML (Figure 3.5a), where a triangle connects a superclass to its subclasses. Generalizations can be specified as *total* or *partial*. A generalization is total when the union of all instances of the subclasses is equivalent to the complete set of instances of the superclass. They can also be *disjoint* or *overlapping*. In a disjoint generalization/specialization, an instance must belong to at most one subclass.

Conceptual generalization allows modeling objects with multiple geographic representations, which may vary according to the scale or to the geometric shape. In this type of relationship, the superclass does not have a geographic representation, but each subclass can have a different representation. Alphanumeric attributes defined in the superclass are inherited by the subclasses. The representation and attributes of the subclasses are also included, according to their intended representation scale (or range of scales), or they can be taken as an alternative way to represent the same object in different contexts. Instances in subclasses of conceptual generalizations can be either disjoint or overlapping (Figure 3.5b).

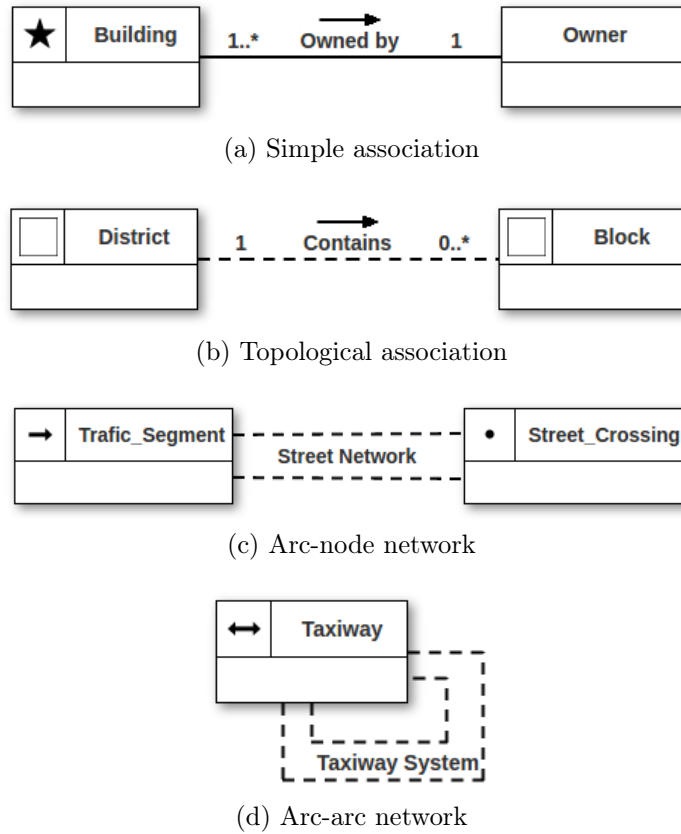


Figure 3.4: OMT-G notations for relationships.

Aggregation is a special form of association between objects, where one of them is considered to be congregated from others. The graphic notation used in OMT-G follows the one used in UML (Figure 3.5c). An aggregation can occur between conventional classes, georeferenced classes and georeferenced and conventional classes. In the latter case, a *spatial aggregation* (i.e. “whole-part” aggregations) must be used (Figure 3.5d).

3.2 Integrity Constraints

The OMT-G model allows several spatial integrity rules to be derived from its primitives. These rules constitute a set of constraints that must be observed during insert, update or delete operations of a spatial database [Borges et al., 2002; Davis Jr. et al., 2001, 2005]. Some spatial integrity constraints are defined implicitly as part of the semantics of the primitives. Other constraints can be deduced from the schemas. Spatial integrity constraints are defined for topological relationships, network relationships, spatial aggregation and geo-field classes. User-defined integrity constraints can also be created by specifying business rules and semantic constraints in the schema. There

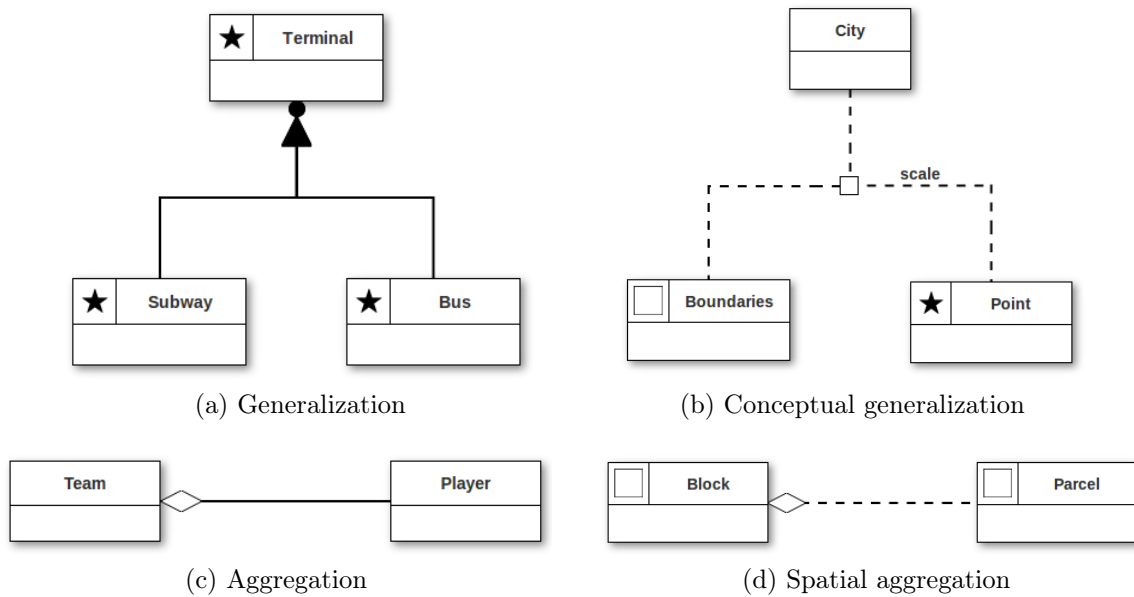


Figure 3.5: OMT-G notations for generalizations and aggregations.

are also constraints which apply to the geometric representation of geo-objects (i.e., constraints related to the consistency of points, lines, polygons etc.).

3.2.1 Geo-field constraints

The spatial integrity rules (R1 – R5) can be deduced from the semantics involved in the concept of geo-fields and, also, from the specific definition of its four descendant classes. Particularly, all representations must correspond to the entire field being modeled, that is, it must be possible to infer a value at any point of the field of interest from the information contained in one of these representations.

(R1) Planar Enforcement Rule: Let F be a geo-field and let P be a point such that $P \subset F$. Then a value $V(P) = f(P, F)$, i.e., the value of F at P , can be univocally determined.

(R2) Isoline: Let F be a geo-field. Let v_0, v_1, \dots, v_n be $n + 1$ points in the plane. Let $a_0 = \overline{v_0 v_1}$, $a_1 = \overline{v_1 v_2}$, \dots , $a_{n-1} = \overline{v_{n-1} v_n}$ be n segments, connecting the points. These segments form an *isoline* L if, and only if, (1) the intersection of adjacent segments in L is only the extreme point shared by the segments (i.e., $a_i \cap a_{i+1} = v_{i+1}$), (2) non-adjacent segments do not intercept (that is, $a_i \cap a_j = \emptyset$ for all i, j such that $j \neq i + 1$), and (3) the value of F at every point P such that $P \in a_i$, $0 \leq i \leq n$, is constant.

- (R3) Tessellation:** Let F be a geo-field. Let $C = \{c_0, c_1, c_2, \dots, c_n\}$ be a set of regularly-shaped cells covering F . C is a tessellation of F if and only if for any point $P \subset F$, there is exactly one corresponding cell $c_i \in C$ and, for each cell c_i , the value of F is given.
- (R4) Planar Subdivision:** Let $A = \{A_0, A_1, A_2, \dots, A_n\}$ be a set of polygons and F be a geo-field. Such that $A_i \subset F$ for all i such that $0 \leq i \leq n$. A forms a planar subdivision representing F if and only if for any point $P \subset F$, there is exactly one corresponding polygon $A_i \in A$, for which a value of F is given (that is, the polygons are non-overlapping and cover F entirely).
- (R5) Triangular Irregular Network (TIN):** Let F be a geo-field. Let $T = \{T_0, T_1, T_2, \dots, T_n\}$ be a set of triangles such that $T_i \subset F$ for all i such that $0 \leq i \leq n$. T forms an triangular irregular network representing F if and only if for any point $P \subset F$, there is exactly one corresponding triangle $T_i \in T$, and the value of F is known at all of vertices of T_i .

3.2.2 Geo-object constraints

The geometric concepts used in the definition of points, lines (including lines with a topological role), and polygons lead to some integrity constraints. In OMT-G it is admissible the existence of geo-objects that are formed by multiple polygons, establishing one of them as a base polygon and considering the others as holes. The following constraints (R6 – R8) are regarding lines and polygons.

- (R6) Line:** Let v_0, v_1, \dots, v_n be $n + 1$ points in the plane. Let $a_0 = \overline{v_0v_1}$, $a_1 = \overline{v_1v_2}$, \dots , $a_{n-1} = \overline{v_{n-1}v_n}$ be n segments, connecting the points. These segments form a *polygonal line* L if, and only if, (1) the intersection of adjacent segments in L is only the extreme point shared by the segments (i.e., $a_i \cap a_{i+1} = v_{i+1}$), (2) non-adjacent segments do not intercept (that is, $a_i \cap a_j = \emptyset$ for all i, j such that $j \neq i + 1$), and (3) $v_0 \neq v_n$, that is, the polygonal line is not closed.
- (R7) Simple Polygon:** Let v_0, v_1, \dots, v_n be $n + 1$ points in the plane, with $n > 3$. Let $s_0 = \overline{v_0v_1}$, $s_1 = \overline{v_1v_2}$, \dots , $s_{n-1} = \overline{v_{n-1}v_n}$ be a sequence of $n - 1$ segments, connecting the points. These segments form a *simple polygon* P if, and only if, (1) the intersection of adjacent segments in P is only the extreme point shared by the segments (i.e., $s_i \cap s_{i+1} = v_{i+1}$), (2) non-adjacent segments do not intercept (i.e., $s_i \cap s_j = \emptyset$ for all i, j such that $j \neq i + 1$), and (3) $v_0 = v_n$, that is, the polygon is closed.

(R8) Polygonal Region: Let $R = \{P_0, P_1, \dots, P_{n-1}\}$ be a set formed by n simple polygons in the plane, with $n > 1$. Considering P_0 to be a basic polygon, R forms a *polygonal region* if, and only if, (1) polygon P_0 has its vertices coded in a counterclockwise fashion, (2) P_i disjoint P_j for all $P_i \neq P_0$ in which the vertices are coded counterclockwise, and (3) P_0 contains P_i for all $P_i \neq P_0$ in which the vertices are coded clockwise.

3.2.3 Network relationship constraints

Network relationships involve objects that are connected with each other. These relationships only show the need for a logical connection, not requiring the implementation of any particular data structure. The connectivity rules, which apply to network relationship primitives, are R9 and R10.

(R9) Arc-node network: Let $G = \{N, A\}$ be a network structure composed of a set of nodes $N = \{n_0, n_1, \dots, n_p\}$ and a set of arcs $A = \{a_0, a_1, \dots, a_q\}$. Members of N and members of A are related according to the following constraints:

1. For every node $n_i \in N$ there must be at least one arc $a_k \in A$.
2. For every arc $a_k \in A$ there must be exactly two nodes $n_i, n_j \in N$.

(R10) Arc-arc network: Let $G = \{A\}$ be a network structure composed of a set of arcs $A = \{a_0, a_1, \dots, a_q\}$. Then the following constraint applies:

1. Every arc $a_k \in A$ must be related to at least one other arc $a_i \in A$, where $k \neq i$.

3.2.4 Spatial aggregation constraint

Aggregation is a special form of association between objects, where one of them is considered to be mounted from others and can occur between all types of classes. When the aggregation is between georeferenced classes, a spatial integrity constraint is imposed considering the existence of the aggregated object and its corresponding sub-objects. This constraint must verify that the geometry of the whole is fully covered by the geometry of the parts and that no overlapping among the parts occurs, as described in rule R11.

(R11) Spatial aggregation: Let $P = \{P_0, P_1, \dots, P_n\}$ be a set of geo-objects. Then P forms another object W by spatial aggregation if, and only if, (1) $P_i \cap W = P_i$

for all i such that $0 \leq i \leq n$, and (2) $(W \cap \bigcup_{i=0}^n P_i) = W$, and (3) $((P_i \text{ touch } P_j) \vee (P_i \text{ disjoint } P_j)) = \text{TRUE}$ for all i, j such that $i \neq j$.

3.2.5 Spatial relationship constraints

Spatial relations represent *direction*, *topological*, *metric*, *ordinal*, and *fuzzy* relationships. Some relationships can be derived from the geometry of each object, during the execution of data insertion or spatial analysis operations. Others need to be specified by the user, in order to allow the system to store and maintain that information. OMT-G considers a set of five basic spatial relationships between georeferenced classes, from which all others can be derived [Clementini et al., 1993; Davis Jr. et al., 2005]: *crosses*, *disjoint*, *overlaps*, *touches* and *within*.

The integrity constraints RT1 to RT5 consider these spatial relationships types. These constraints are formulated using a notation in which objects are indicated by upper-case italic letters (e.g., A , B), their boundaries are denoted as ∂A , and their interiors as A° . The boundary of a point object is considered to be always empty (therefore, the point is equivalent to its interior), and the boundary of a line is consisted of its two endpoints. A function, named *dim*, is used to return the dimension of an object and returns 0 if the object is a point, 1 if it is a line, or 2 if it is a polygon. Notice that some relationships are only allowed between specific classes because they depend on the geometric representation. For instance, the existence of a *touches* relationship assumes that none of the classes involved are a point.

(RT1) Crosses: Let A be a geo-object of the **Line** class, and let B be a geo-object of either the **Line** or the **Polygon** class. Then $(A \text{ crosses } B) = \text{TRUE} \Leftrightarrow \dim(A^\circ \cap B^\circ) = (\max(\dim(A^\circ), \dim(B^\circ)) - 1) \wedge (A \cap B \neq A) \wedge (A \cap B \neq B)$.

(RT2) Disjoint: Let A and B be two geo-objects. Then $(A \text{ disjoint } B) = \text{TRUE} \Leftrightarrow A \cap B = \emptyset$.

(RT3) Overlaps: Let A and B be two geo-objects. Both members of the **Line** or of the **Polygon** class. Then $(A \text{ overlaps } B) = \text{TRUE} \Leftrightarrow \dim(A^\circ) = \dim(B^\circ) = \dim(A^\circ \cap B^\circ) \wedge (A \cap B \neq A) \wedge (A \cap B \neq B)$.

(RT4) Touches: Let A and B be two geo-objects, where neither A nor B are members of the **Point** class. Then $(A \text{ touches } B) = \text{TRUE} \Leftrightarrow (A^\circ \cap B^\circ = \emptyset) \wedge (A \cap B \neq \emptyset)$.

(RT5) Within: Let A and B be two geo-objects, where B is an instance of the class **Polygon**. Then $(A \text{ within } B) = \text{TRUE} \Leftrightarrow (A \cap B = A) \wedge (A^\circ \cap B^\circ \neq \emptyset)$.

OMT-G considers, for convenience, a larger set of spatial relationships constraints, due to cultural or semantic concepts that are familiar to the users. These constraints (RT6 to RT13) are special cases of one of the five basic relationships, but they deserve a special treatment because of their common use in practice. Constraints RT12 and RT13 represent metric relationships and require a parameter, since the notion of proximity varies according to the situation, a precise distance must be supplied in order to allow the correct evaluation of the relationship.

(RT6) Equals: Let A and B be two geo-objects. Then $(A \text{ equals } B) = \text{TRUE} \Leftrightarrow A \cap B = A = B$.

(RT7) Contains: Let A and B be two geo-objects, where A is a member of the **Polygon** class. Then $(A \text{ contains } B) = \text{TRUE} \Leftrightarrow ((B \text{ within } A) = \text{TRUE}) \wedge ((A \text{ equals } B) = \text{FALSE})$.

(RT8) ContainsProperly: Let A and B be two geo-objects, where A is a member of the **Polygon** class. Then $(A \text{ containsproperly } B) = \text{TRUE} \Leftrightarrow ((B \text{ within } A) = \text{TRUE}) \wedge ((A \text{ touches } B) = \text{FALSE})$.

(RT9) Covers: Let A and B be two geo-objects. Then $(A \text{ covers } B) = \text{TRUE} \Leftrightarrow A \cap B = A$.

(RT10) CoveredBy: Let A and B be two geo-objects. Then $(A \text{ covers } B) = \text{TRUE} \Leftrightarrow A \cap B = B$.

(RT11) Intersects: Let A and B be two geo-objects. Then $(A \text{ intersects } B) = \text{TRUE} \Leftrightarrow ((A \text{ disjoint } B) = \text{FALSE})$.

(RT12) Distant(dist): Let A and B be two geo-objects. Let C be a buffer, created at a distance $dist$ around A . Then $(A \text{ distant}(dist) B) = \text{TRUE} \Leftrightarrow (B \text{ disjoint } C) = \text{TRUE}$.

(RT13) Near(dist): Let A and B be two geo-objects. Let C be a buffer, created at a distance $dist$ around A . Then $(A \text{ near}(dist) B) = \text{TRUE} \Leftrightarrow (B \text{ disjoint } C) = \text{FALSE}$.

Table 3.1: Geometric types: OMT-G and OGC SFS

OMT-G representation	OGC SFS representation
Point	Point
Line	LineString
Polygon	MultiPolygon
Node	Point
Unidirectional Arc	LineString
Bidirectional Arc	LineString
Isolines	LineString
Sample	Point
Planar Subdivision	Polygon or Multipolygon
Triangular Irregular Network	Point (vertices) and Polygon (triangles)
Tessellation	–

3.3 Mapping OMT-G to object-relational spatial databases

Geometric types in conceptual schemas are accompanied by an expected behavior, captured with a set of spatial integrity constraints [Borges et al., 2002]. Geo-objects, for example, like *lines*, *arcs* and *polygons* must be formed by simple lines or simple polygonal lines, that is, lines without self intersection or self tangency. Geo-fields, like *sampling*, *tessellation*, *planar-subdivision*, *isoline* and *triangular irregular network* must be continuously distributed over the space, without overlapping between adjacent lines or polygons.

In contrast, the data types implemented by most modern spatial RDBMS are simple geometry types and geometry collections hierarchized by OGC. When mapping a spatial data model from the conceptual to the physical schema, we are forced to use these simple geometric representations available in the spatial RDBMS. This process implies in loss of semantics, since the only topological constraints implemented in the spatial RDBMS are simple value checks (e.g. a polygon is a closed line). Such constraints can ensure the geometric consistency of objects represented by lines or polygons. However, ensuring the consistency of aggregations or arc-node relationships, for example, is more complicated, usually requiring the development of triggers. This implementation, however, is not trivial and demands advanced knowledge of the resources offered by the RDBMS.

Table 3.1 exemplifies how the OMT-G primitives are mapped to the OGC representations. Take for instance the geometries *line*, *unidirectional arc*, *bidirectional arc*

and *isolines*. They have completely different behavior in the conceptual model, but they are all mapped to *linestring* in the physical schema.

3.4 Further work on OMT-G

OMT-G motivated many initiatives, including Wispy [Fatto et al., 2015], a uDig¹ extension that permits verifying and visually specifying complex spatial integrity constraints. WiSPY includes a visual environment for defining spatial data models with integrity constraints and for automatically generating the constraint checker. The latter is used to verify the integrity of the data produced during the map editing process.

In an earlier work, Borges et al. [2005] inaugurate the study of spatial integrity constraints from OMT-G schemas, and propose an algorithm that allows for the mapping between an OMT-G class diagram and an object-relational schema, which includes basic geometric representations as part of relations, along with conventional attributes. A list of conventional and spatial integrity constraints is also obtained. From the object-relational schema, a physical schema for spatially extended relational databases is easily derived, but spatial integrity constraints must be implemented using triggers, checks and assertions.

Hora et al. [2010] later implemented an OMT-G mapping tool to generate Oracle PL/SQL schemas², that includes triggers and procedures, and XML schemas³. In [Hora et al., 2011], they proposed a methodology and an algorithm to map arcs and nodes, organized in a network using spatial relationships, from a OMT-G schema to a GML document.

Lizardo and Davis Jr. [2014] presented the OMT-G Designer⁴, a web-based modeling tool for OMT-G that includes Hora et al.’s mapping algorithms. In this work, we extend OMT-G Designer with an alternative mapping algorithm for PostgreSQL/-PostGIS, that includes the spatial integrity constraints and advanced spatial data types introduced by AST-PostGIS.

Finally, Seufitelli et al. [2015] identified the challenges in mapping OMT-G primitives for non-relational paradigms in order to integrate relational and non-relational databases, creating a hybrid approach.

¹uDig: <http://udig.refractory.net/>

²OMTG2SQL: <https://github.com/lab-csx-ufmg/omtg2sql>

³OMTG2GML: <https://github.com/lab-csx-ufmg/omtg2gml>

⁴OMT-G Designer: <http://aqui.io/omtg/>

Chapter 4

AST-PostGIS

In this work, we propose AST-PostGIS (Advanced Spatial Types for PostGIS), an open-source SQL extension that implements conceptual design semantics for spatial relational database management systems. Written in PL/pgSQL¹, AST-PostGIS is currently available for PostgreSQL version 9.5 or superior and requires the spatial module PostGIS, version 2.0 or above. Like any other PostgreSQL extension, AST-PostGIS is easy to install and can be individually enabled in each database schema. During installation, the extension creates several new data types, functions, procedures and a table. In order to discern these extension objects from those already implemented in PostgreSQL and PostGIS, AST-PostGIS adopts as a standard the **advanced spatial type (AST)** prefix for all names.

AST-PostGIS is intended to bridge the gap between the conceptual design and the physical implementation of spatial databases. By introducing advanced spatial data types, AST-PostGIS allows creating geographic columns on tables with behavior control, respecting the semantics for geo-objects and geo-field geometries defined by OMT-G. By installing trigger procedures to assert the consistency of spatial relationships during data updates, AST-PostGIS permits making explicit roles for spatial relations, as, for example, network connectivity. Furthermore, by providing functions to verify the consistency of the database before enforcing relationships constraints, the extension allows to sanitize the data input in bulk. Those functions manage all the necessary information to identify inconsistent data, and indicate constraint violations in a log table.

The following three sections explain, in detail, each of these three features introduced by AST-PostGIS.

¹PL/pgSQL is a loadable procedural language for the PostgreSQL database system.

Table 4.1: Advanced spatial data types supported by AST-PostGIS

Advanced Spatial Types	OMT-G Class	PostGIS Primitive	Integrity Constraints
ast_point	point	geometry(point)	-
ast_line	line	geometry(linestring)	R6
ast_polygon	polygon	geometry(multipolygon)	R7, R8
ast_node	node	geometry(point)	-
ast_isoline	isoline	geometry(linestring)	R1, R2
ast_planar subdivision	planar subdivision	geometry(polygon)	R1, R4
ast_tin	triangular irregular network	geometry(polygon)	R1, R5
ast_tessellation	tessellation	raster	R1, R3
ast_sample	sample	geometry(point)	-
ast_uniline	unidirectional line	geometry(linestring)	R6
ast_biline	bidirectional line	geometry(linestring)	R6

4.1 Advanced Spatial Data Types

Advanced Spatial Data Types are essentially the primitive geometric types of PostGIS coupled with a set of spatial integrity constraints to control their behavior, as expected by the designer in the conceptual level. These new spatial data types can be handled in the same way primitive types are, as they can be employed as column definition of tables, as variables in PL/pgSQL scripts or as arguments of functions or stored procedures. They can also be stored, retrieved and updated with the geometry processing functions of PostGIS.

The AST-PostGIS extension implements 11 Advanced Spatial Data Types. They are derived from the concepts of geo-objects and geo-fields classes of the OMT-G model and their semantics are controlled by the integrity constraints R1 – R8, defined formally in Section 3.2. These integrity constraints were implemented in PL/pgSQL scripts and encapsulated in the extension. Table 4.1 lists the Advanced Spatial Data Types implemented in AST-PostGIS, along with their georeferenced classes of the OMT-G model from where they were derived, the PostGIS primitive types that correspond to their geometry, and the integrity constraints that control their behavior.

4.2 Integrity constraints for spatial relationships

AST-PostGIS provides integrity constraints to spatial relationships through using triggers. When fired, these triggers must execute custom procedures introduced by the extension. AST-PostGIS provides three procedures: *ast_spatialrelationship*, *ast_arcnodenetwork* and *ast_aggregation* that cover, respectively, spatial relationships, arc-node networks and spatial aggregations. All three derive from OMT-G primitives. In addition, *ast_arcnodenetwork* can also implement integrity constraints for arc-arc networks.

In PostgreSQL, a trigger is associated with only one table and it executes a procedure when a certain event occurs in this table. Events can be either an **insert**, an **update** or a **delete** operation. Triggers can be specified to fire **before** events are attempted or **after** the events have been completed. In the latter situation, the state of the database is evaluated after the event completion and, in case the trigger constraints are violated, an exception is raised and a rollback operation is performed. If a trigger is marked **for each row**, it is called once for every row that the operation modifies, but a trigger that is marked **for each statement**, only executes once for any given operation, regardless of how many rows it modifies.

To implement integrity constraints on spatial relationships, AST-PostGIS requires that a trigger execute one of the custom procedures and be configured to fire **after** an event occur and be marked with **for each statement**. The table associated with the trigger must also be one of the tables involved in the relationship. In case of a spatial aggregation relationship, the table associated must be the one that represents the *part* of the relationship. For arc-node relationships, the associated table can be either the *arc* or the *node* of the relationship. If it is the arc, the trigger blocks arc insertions or updates if there are no two nodes connected to them. If the associated table is a node, the trigger blocks nodes not connected to any arc. Spatial relationship triggers can also be associated to both tables of the relationship, but the choice of the table changes how the constraints are applied to the relationship.

The type of the operation must also be chosen according to the relationship on which the trigger is being applied. Arc-node networks and spatial relationships require the trigger to fire after **insert** and **update** operations, while spatial aggregations demand all three operations (**insert**, **update** and **delete**).

Besides having the name of the table associated directly with the trigger, the names of both tables must also be passed to the procedure as a parameter. Although this requirement is a bit redundant, it is necessary for the trigger to identify which is the other table involved in the relationship. Furthermore, as feature tables can have multiple geometric columns, the name of these columns, associated in the relationships, must be passed to the procedure to avoid ambiguity. The order of the parameters is also important. For example, spatial aggregation trigger requires the *part* table name to be passed as the first parameter along with its geometric column name as the second parameter. Followed by the *whole* table name, as third parameter, and its the geometric column name, as fourth parameter. Listing 4.1 shows how a trigger for a spatial aggregation can be created.

The procedure `ast_arcnodenetwork` is polymorphic and admits two configurations for the parameters. When used with an arc-node network, the procedure requires four

```

CREATE TRIGGER trigger_name
  AFTER INSERT OR UPDATE OR DELETE
  ON part_tbl
  FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_aggregation(
      'part_tbl', 'part_geom',
      'whole_tbl', 'whole_geom'
    );

```

Listing 4.1: Aggregation Trigger

```

CREATE TRIGGER trigger_name
  AFTER INSERT OR UPDATE
  ON [ arc_tbl | node_tbl ]
  FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_arcnodenetwork(
      'arc_tbl', 'arc_geom',
      'node_tbl', 'node_geom'
    );

```

Listing 4.2: Arc-Node Trigger

parameters, which are (1) the name of arc table, (2) the name of geometric column (must be of type *ast_uniline* or *ast_biline*), (3) the name of table representing a node, and (4) the name of the column of the node geometry, whose type is *ast_node*. The configuration of a trigger and its procedure to a arc-node network is illustrated in Listing 4.2. When the procedure is used with an arc-arc network relationship, only two parameters are accepted, which are (1) the arc table’s name and (2) its geometric column.

AST-PostGIS considers a set of 12 different spatial relationships between two georeferenced classes. What differentiates one type of relationship from another is the relationship operator passed to the trigger procedure.

AST-PostGIS supports the minimum set of spatial relationship operators, identified by Clementini et al. [1993] and adopted by OMT-G [Davis Jr. et al., 2005], from which all others can be specified: **Crosses**, **Disjoint**, **Overlaps**, **Touches** and **Within**. Besides them, the extension considers the larger set of spatial relationships (rules RT, described in Section 3.2.5) for convenience. This includes relationships such as **Contains**, **ContainsProperly**, **Covers**, **CoveredBy** and **Intersects**, which are in fact special cases of the five basic relationships, but due to the common use in practice, they should be available along with the minimum set. In addition, the two metric

```

CREATE TRIGGER trigger_name
  AFTER INSERT OR UPDATE
  ON a_table
  FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_spatialrelationship(
      'a_table', 'a_geom',
      'b_table', 'b_geom',
      'spatial_relationship', <'distance'>
    );

```

Listing 4.3: Spatial Relationship Trigger

spatial relationships are admitted: **Distant** and **Near**. These require as a parameter the value of the distance in which the relationship occurs.

The parameters passed to *ast_spatialrelationship* are: (1) table A name, (2) geometry A name, (3) table B name, (4) geometry B name, and (5) spatial relationship operator. When the spatial relationship is **Distant** or **Near**, a sixth parameter is admitted with the value of the distance. Listing 4.3 illustrates the trigger configuration for a topological relationship. This trigger requires to be associated to table A, whose name is passed as a parameter after the statement *ON*.

In PostgreSQL, triggers are only fired if an event ensues on the table to which they are associated. A problem arises when users alter the other table of the relationship, if there is no trigger on that table to catch the event. This operation can lead the spatial relationship to an inconsistent state. For instance, consider an arc-node relationship on which the trigger was created associated to the arc table. This trigger ensures that every arc instance, stored in the table, is connected to two nodes from the node table. However, if a user deletes a node on the node table connected to an arc, no trigger would catch and block this operation, leaving an arc connected to only one node. This situation violates rule R9, defined in Section 3.2.

The aforementioned problem is addressed by AST-PostGIS by creating a second trigger associated to the other table of the relationship. This auxiliary trigger is implemented automatically, when the trigger written by the user is created. This trigger is configured to fire only after **delete** events for arc-node networks and spatial relationships, and **update** and **delete** events for spatial aggregations.

Table 4.2: Consistency check functions supported by AST-PostGIS

Spatial relationship	Consistency check function
Spatial Relationship	<code>ast_isSpatialRelationshipValid (a_tbl text, a_geom text, b_tbl text, b_geom text, relationship text)</code> <code>ast_isSpatialRelationshipValid (a_tbl text, a_geom text, b_tbl text, b_geom text, relationship text, dist real)</code>
Arc-Node Network	<code>ast_isNetworkValid (arc_tbl text, arc_geom text, node_tbl text, node_geom text)</code>
Arc-Arc Network	<code>ast_isNetworkValid (arc_tbl text, arc_geom text)</code>
Spatial Aggregation	<code>ast_isSpatialAggregationValid (part_tbl text, part_geom text, whole_tbl text, whole_geom text)</code>

```
SELECT ast_isNetworkValid( arguments );
```

Listing 4.4: Usage of the consistency check function for network relationship

4.3 Consistency Check Functions

The spatial integrity constraints introduced in the previous subsection have to be applied when a database is created and before any data is stored. They work by checking data operation events (like insertions, updates and deletions) when they occur. To verify a non empty spatial database for violations on spatial relationships, AST-PostGIS provides consistency check functions, as shown in Table 4.2. These functions can be called before the initial enforcement of constraints, and they not only inform if the spatial relationship is invalid, but also identify the geometries that cause the violation.

The consistency check functions are not executed by triggers. Instead, they are called by a straightforward SELECT statement, omitting the FROM clause as shown in Listing 4.4. Just like the trigger procedures described in last subsection, consistent check functions also require parameters, which are the same as in the procedures.

The `ast_violation_log` table is used by the consistency check functions to record information about the inconsistency encountered on the the spatial relationships. The schema of the table is shown in Figure 4.1. This table was designed to support the necessary information for the recovery of inconsistencies. It stores the type of relationship that was violated, a description with information about the rows related to the error, and also the geometry that causes the error.

ast_violation_log
time: TIMESTAMP
type: VARCHAR(50)
description: VARCHAR(150)
geomAsText: TEXT

Figure 4.1: AST-PostGIS log table schema

4.4 Summary of limitations

AST-PostGIS has three known limitations, mostly due to the way spatial representations and relationships are implemented in a RDBMS extension.

First, the triggers that create integrity constraints for relationships require a rigid statement structure in their creation. The triggers must be specified with different statements for each relationship type, as explained in Section 4.2. This is necessary to overcome the limitations imposed by PostgreSQL/PostGIS in the support for spatial concepts. It would be simpler, instead, if integrity constraints for spatial relationships, networks and aggregations could be specified using the DDL, analogously to the FOREIGN KEY statement for referential integrity in SQL, with the necessary verifications included in the DBMS's code. In order to overcome this problem, AST-PostGIS runs scripts to check if the trigger statements are correct during their creation processes, and raises clarifying exceptions if not.

The trigger procedures, introduced by our extension to add integrity constraints on spatial relationships, receive as parameters not only the names of the feature tables involved in the relationship, but also the names of their corresponding geometric columns. These parameters are required to avoid ambiguity problems when feature tables have multiple georeferenced columns, although in most cases there is only one geometric column per table. Using the *geometry_columns* view in PostGIS does not resolve this problem.

The third limitation regards the lack of spatial boundaries for geo-fields in OMT-G. Without an indication of the limits of the space of interest for the application, AST-PostGIS cannot adequately check the planar enforcement rule (R1). OMT-G conceptual schemas that involve geo-fields should (as a good practice) include a class to represent a frame of reference for the application's spatial limits, but the model does not include any primitive for that purpose.

Chapter 5

Case Study: Urban geographic database

In order to illustrate the use of AST-PostGIS, we present in this section an implementation of a small spatial database schema in PostgreSQL/PostGIS, using the advanced spatial types and integrity constraints described in the previous chapter. The schema was composed in the online interactive design tool, OMT-G Designer [Lizardo and Davis Jr., 2014] that is capable of automatically mapping the OMT-G diagram to PostGIS using AST-PostGIS, generating a complete set of DDL commands and triggers in a script.

In Section 5.1, we introduce the conceptual model for an urban cadastral database system and explain the semantics and spatial integrity constraints that can be observed in the example. Next, in Section 5.2, we show how this example is mapped to the physical schema using the elements of our SQL extension. Finally, in Section 5.3, we present the responses of AST-PostGIS to data updates that violate spatial integrity constraints.

5.1 Conceptual schema

Figure 5.1 shows a conceptual schema fragment of an urban cadastral database system. Its class diagram includes most of the primitives defined in OMT-G and we can observe several spatial integrity constraints in it.

The schema fragment corresponds to the geographic area of a municipality. The city can be represented as a point or as a polygon, by its boundaries. These boundaries contain a number of blocks, which are in turn subdivided into parcels. Each parcel is represented by its polygonal boundary and can be occupied by residence, commerce

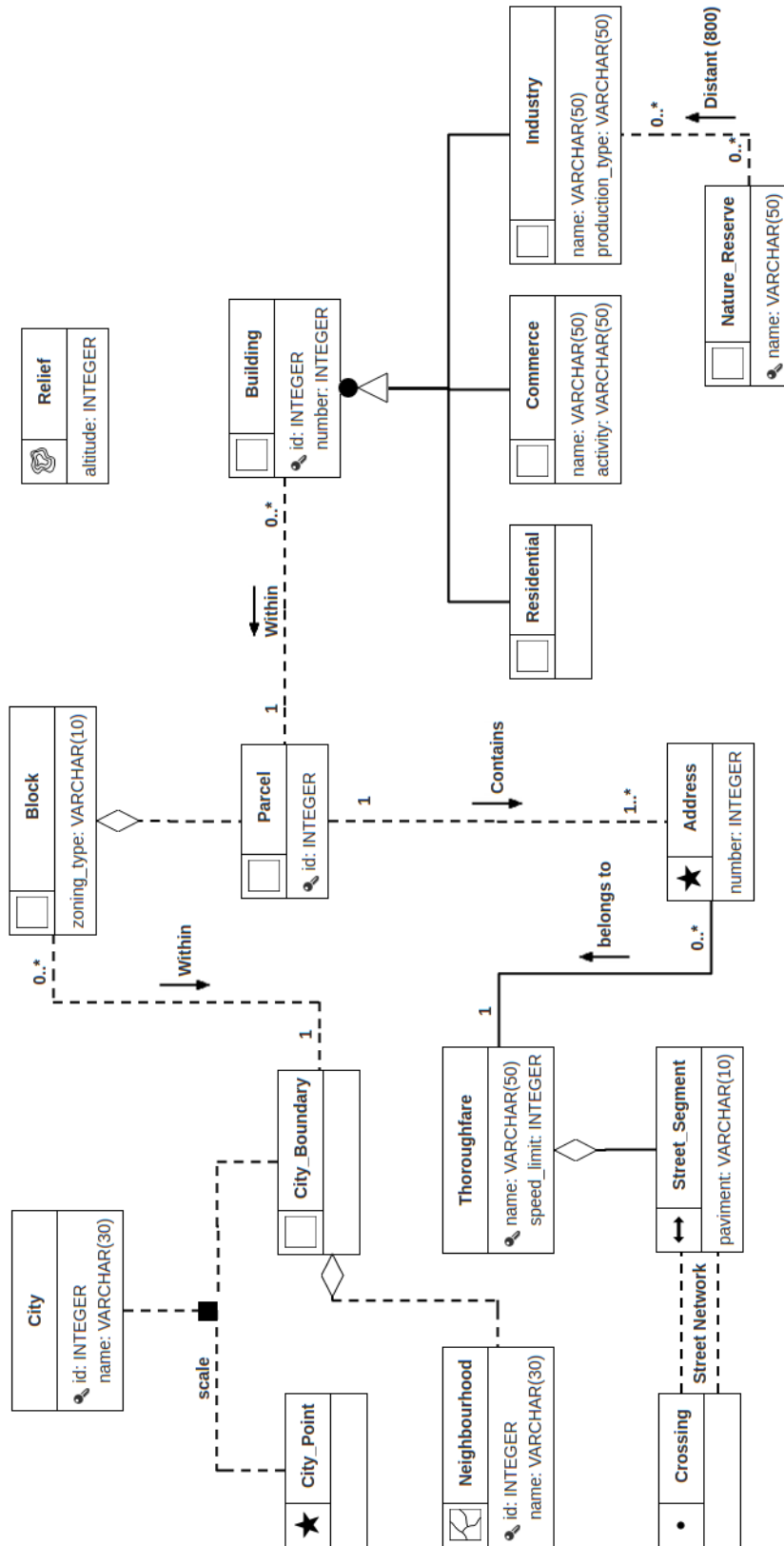


Figure 5.1: OMT-G schema fragment for an urban geographic application

and industry buildings. Due to environmental regulations, industries cannot be built within 800 meters of nature reserves. Building addresses are formed by concatenating the thoroughfare name to the street number. Each address is defined as a symbol, and is to be located inside the parcel area. A thoroughfare is modeled as an aggregation of street segments, thereby composing the arcs in a street network. Thoroughfare intersections are represented by nodes at the crossings. The municipality space is also entirely subdivided into neighborhoods. In addition, relief is represented by a set of isolines, a geo-field that covers the whole municipal territory.

Several spatial integrity constraints can be derived from the semantics of this schema. Spatial aggregation primitives have been used in the relationship between city boundaries and neighborhoods, and between blocks and parcels. This rule ensures that each parcel, for example, must be contained in only one block instance. Parcels must be adjacent to the other, ensuring no overlapping and no empty spaces. In addition, parcel's boundaries must be entirely contained inside the block's boundary. This means, instances from blocks and parcels must be of the same size and must exist in the same location. Same rules are applied between city boundaries and neighborhoods.

Spatial relationship integrity constraints are also seen in this example. Block areas must be inside the city boundaries. Parcels' polygonal areas contain addresses and buildings, which can either be residential, commerce or industry. Between industries and nature reserves we notice a special type of topological relationship constraint. In this case, industries locations must be farther than 800 meters from nature reserves instances. A primitive of a network relationship is used to represent the street network. This primitive encapsulates the cardinality of the relationship and defines that each street must be connected to two nodes, represented by the *Crossing* class.

The integrity constraints that must be observed in the physical implementation are summarized in Table 5.1.

5.2 Physical implementation

The SQL snippet in Listing 5.1 shows how part of the conceptual schema illustrated in Figure 5.1 can be mapped to a PostgreSQL/PostGIS database. In this example, all tables, but *Thoroughfare*, are georeferenced and have a geometric column declared with the advanced spatial types supported by AST-PostGIS. As *City* can be represented either as a point or as a polygon, its table was created with two geometric columns: *geom_point* and *geom_boundary*. Due to a conventional association and a conventional aggregation, tables *Address* and *Street_Segment* have foreign keys referencing the table

Table 5.1: Spatial integrity constraints derived from the schema in Figure 5.1

Rule	Classes
R1	Relief, Neighborhood
R2	Relief
R4	Neighborhood
R6	Street segment
R7/R8	City boundary, Block, Parcel, Building, Nature reserve
R9	Street Network (Crossing/Street segment)
R11	Block/Parcel
RT	Block <i>within</i> City boundary, Building <i>within</i> Parcel, Parcel <i>contains</i> Address, Nature reserve <i>distant</i> (800) Industry

Thoroughfare. The geometric columns of these two tables are of types *ast_point* and *ast_biline*, respectively. To keep the example concise, we refrain from showing the code necessary to construct the indexes on the geometric columns of each relation, but the complete SQL schema is available in Appendix A.

The code snippet in Listing 5.2 illustrates how integrity constraints are implemented for the two spatial aggregation primitives presented in the conceptual schema. A trigger for each aggregation is created firing the *ast_aggregation* procedure after every insertion, update or deletion of data on *Parcel* and *Neighborhood* tables. Both tables represent the *Part* of the “Whole-Part” aggregation. This procedure receives as parameters the names of the two tables involved in the spatial aggregation together with their geometric columns. In the case of the aggregation between tables *City* and *Neighborhood*, the *geom_boundary* column of *City* is passed to the procedure as a parameter. Those triggers ensure no overlaps between parcels and neighborhoods and guarantee that the geometry of blocks and the geometry of city boundaries will be fully covered by the geometry of the parcels and neighborhoods, respectively.

In a similar way, spatial integrity constraints for spatial relationships are also implemented with triggers. In this case, the procedure called is *ast_spatialrelationship*, which receives as parameters the name of the two tables involved in the relationship along with their geometric column. Moreover, the name of the spatial relationship operator is also passed to the procedure as the fifth parameter. In this example, the spatial relationship ‘*within*’ is passed to the relationship between industries and parcels and between blocks and cities; ‘*contains*’ is passed to the spatial relationship between parcels and addresses; and ‘*distant*’ is passed to the relationship between nature reserves and industries. The spatial relationship ‘*distant*’ requires the value of

```

-- Table City
CREATE TABLE City (
    id integer PRIMARY KEY,
    name varchar(30),
    geom_point ast_point,
    geom_boundary ast_polygon
);
-- Conventional table Thoroughfare
CREATE TABLE Thoroughfare (
    name varchar(50) PRIMARY KEY,
    speed_limit integer
);
-- Table Address
CREATE TABLE Address (
    number integer,
    thoroughfare varchar(50)
        REFERENCES Thoroughfare(name),
    geom ast_point
);
-- Table Street_Segment
CREATE TABLE Street_Segment (
    pavement varchar(10),
    thoroughfare varchar(50)
        REFERENCES Thoroughfare(name),
    geom ast_biline
);

```

Listing 5.1: SQL schema for the tables definition

```

-- Spatial aggregation between Block and Parcel
CREATE TRIGGER aggregation_block_parcel
AFTER INSERT OR UPDATE OR DELETE ON Parcel
FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_aggregation('Parcel', 'geom', 'Block', 'geom');

-- Spatial aggregation between City and Neighborhood
CREATE TRIGGER aggregation_boundary_neighborhood
AFTER INSERT OR UPDATE OR DELETE ON Neighborhood
FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_aggregation('Neighborhood', 'geom', 'City',
        'geom_boundary');

```

Listing 5.2: Integrity constraints implementation for aggregations

```

-- Spatial relationship between Industry and Parcel
CREATE TRIGGER spatial_industry_parcel
AFTER INSERT OR UPDATE ON Industry
FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_spatialrelationship('Industry', 'geom',
        'Parcel', 'geom', 'within');

-- Spatial relationship between Block and City
CREATE TRIGGER spatial_block_city
AFTER INSERT OR UPDATE ON Block
FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_spatialrelationship('Block', 'geom',
        'City', 'geom_boundary', 'within');

-- Spatial relationship between Parcel and Address
CREATE TRIGGER spatial_parcel_address
AFTER INSERT OR UPDATE ON Parcel
FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_spatialrelationship('Parcel', 'geom',
        'Address', 'geom', 'contains');

-- Spatial relationship between Nature and Industry
CREATE TRIGGER spatial_nature_distant
AFTER INSERT OR UPDATE ON Nature_Reserve
FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_spatialrelationship('Nature_Reserve',
        'geom', 'Industry', 'geom', 'distant', '800');

```

Listing 5.3: Integrity constraints implementation for spatial relationships

the distance, which is also passed as a parameter.

Listing 5.3 shows the creation of these triggers for part of the spatial relationships presented in the schema. They ensure, for example, that no industry can be created outside a parcel, or that a parcel cannot be created without an address. Exceptions would be raised by these triggers if a block is not created inside the city boundaries or if an industry is stored in the database without respecting the clearance distant of nature reserves, blocking the invalid data updates. The complete triggers creation for all spatial relationships are available in Appendix A.

Lastly, the trigger in Listing 5.4 implements the spatial integrity constraints for the street network formed by crossing nodes and street segments. The trigger is fired after any insertion or update of data on table *Street_Segment* and execute the procedure *ast_arcnodenetwork*. This procedure ensures that each segment is always connected

```
-- Arc-Node network between Street_Segment and Crossing
CREATE TRIGGER network_street_crossing
AFTER INSERT OR UPDATE ON Street_Segment
FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_arcnodenetwork('Street_Segment', 'geom',
        'Crossing', 'geom');
```

Listing 5.4: Integrity constraint implementation for arc-node network

to two crossings. In case of an inconsistent update or insertion of data, the procedure raises an exception and rolls back the whole operation. As shown in the listing, the procedure receives as parameter the arc and node tables of the network, along with their geometric columns.

5.3 AST-PostGIS responses

Chrisman [1991] argues that error is an integral part of spatial information processing and should be recognized as a fundamental dimension of data. This is because no representation captures a perfect replica of something as complex as the Earth. These forcible deviations between a representation and actual circumstances constitute error. Of course, one goal of any GIS specialist is to avoid needless error. By directly recognizing errors may be possible to confine them to acceptable limits. In this section, we show how AST-PostGIS identifies and avoids errors during data insertions on database that violate the spatial integrity constraints.

Here we present tests that approach three different errors that can occur in a real scenario for the urban cadastral database system modeled in Section 5.1, and implemented in Section 5.2. The first test shows a spatial aggregation mistake, the second is a spatial relationship breach and the third is a arc-node network violation. We explain each situation with 3D illustrations, and present screenshots of the console showing the responses of AST-PostGIS for each case.

Figure 5.2 shows a 3D model of a small urban scenario in which all geographical objects satisfy the integrity constraints defined in the conceptual schema presented in Figure 5.1. For instance, we can notice a street network formed by three crossings. This street network interconnects four urban blocks, which in turn are formed by an aggregation of parcels of different sizes. The parcels are adjacent to each other, without overlapping or gap among them. Inside some of the parcels, there are constructions of houses, industrial sheds and commercial buildings. On the top part of the figure, there is a green linear park, which represents a natural reserve and it is distant from



Figure 5.2: Overview of urban scenario without integrity constraints violations.

the industries located in the bottom left block. Satisfying the clearance restriction between industries and natural reserves.

From this urban scenario, we extract three examples from where a human typo error during a data insertion or even a wrong accuracy measurement could lead to a inconsistent state of the database.

5.3.1 Spatial aggregation violation

The first situation, illustrated in Figure 5.3, is an attempt to register a parcel in the spatial database, whose area extrapolates the limits of the block and overlaps the street line. According to the restriction (*R11*) for spatial aggregation, formalized in Section 3.2.4, the geometry of the block must fully cover the geometries of the parcels. In this manner, no area of parcels can stay outside the block limits.

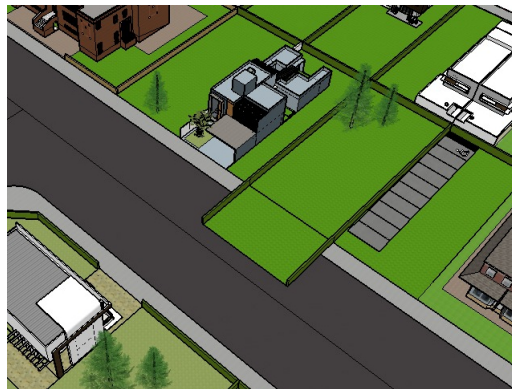
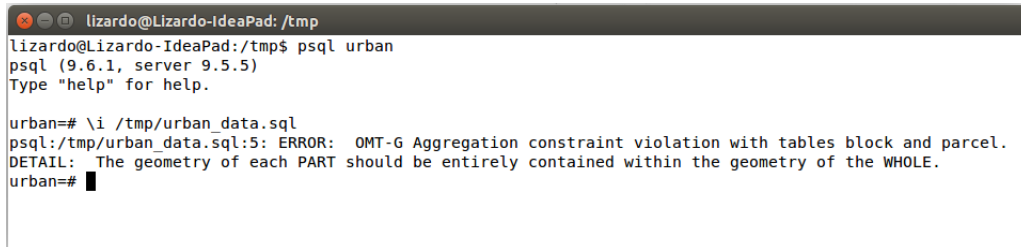


Figure 5.3: Spatial aggregation violation: a parcel extrapolates the block limits.

In an attempt to insert this parcel in a database extended with AST-PostGIS, our extension blocks the transaction and prints an error message explaining the integrity

constraint violation. The error message is shown in detail on the console screenshot illustrated in Figure 5.4.



```

lizardo@Lizardo-IdeaPad: /tmp
lizardo@Lizardo-IdeaPad:/tmp$ psql urban
psql (9.6.1, server 9.5.5)
Type "help" for help.

urban=# \i /tmp/urban_data.sql
psql:/tmp/urban_data.sql:5: ERROR: OMT-G Aggregation constraint violation with tables block and parcel.
DETAIL: The geometry of each PART should be entirely contained within the geometry of the WHOLE.
urban=#

```

Figure 5.4: AST-PostGIS response for a spatial aggregation violation.

5.3.2 Spatial relationship violation

Another situation prone to errors, is when registering the construction area for a building. According to the conceptual schema, every building (regardless of being residential, commercial or industrial) must be constructed within a unique parcel. Figure 5.5 shows a house being registered in the database with area occupying two adjacent parcels.

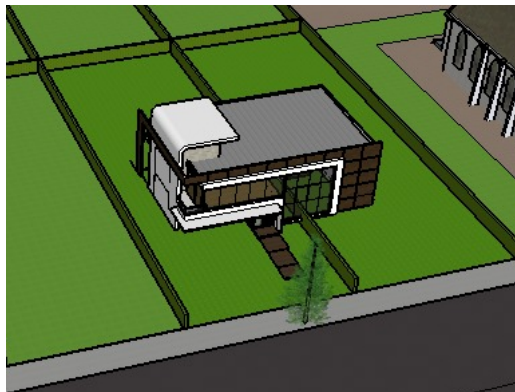


Figure 5.5: Spatial relationship violation: building is not within the parcel area.

AST-PostGIS prevents this error by firing a trigger during data insertion and asserts the spatial integrity constraint (within) defined in the conceptual schema. Figure 5.6 shows the console screenshot with AST-PostGIS report for this violation.

5.3.3 Arc-node network violation

In this last example, we show the AST-PostGIS response for an arc-node network integrity constraint error. According to the restriction (*R9*) of Section 3.2.3, street

```

lizardo@Lizardo-IdeaPad: /tmp
lizardo@Lizardo-IdeaPad:/tmp$ psql urban
psql (9.6.1, server 9.5.5)
Type "help" for help.

urban=# \i /tmp/urban_data.sql
psql:/tmp/urban_data.sql:2: ERROR:  OMT-G Topological Relationship constraint violation (within) between
tables 'residential' and 'parcel'.
urban=# █

```

Figure 5.6: AST-PostGIS response for a spatial relationship integrity constraint violation.

segments (arcs) must intercept exactly two crossings (nodes). Figure 5.7 illustrates this example by showing the street network with a missing crossing.



Figure 5.7: Arc-Node network violation: streets segments (arcs) do not intersect crossings (nodes)

AST-PostGIS identifies this error and reports the error message shown in the console screenshot illustrated in Figure 5.8.

```

lizardo@Lizardo-IdeaPad: /tmp
lizardo@Lizardo-IdeaPad:/tmp$ psql urban
psql (9.6.1, server 9.5.5)
Type "help" for help.

urban=# \i /tmp/urban_data.sql
psql:/tmp/urban_data.sql:2: ERROR:  OMT-G Arc-Node network constraint violation at
table street_segment.
DETAIL:  For each arc at least two nodes must exist at the arc extrem points.
urban=# █

```

Figure 5.8: AST-PostGIS response for a street network integrity constraint violation.

Chapter 6

Conclusions and Future Work

This work introduced AST-PostGIS, an open source PostGIS extension that incorporates advanced spatial data types and implements spatial integrity constraints on a RDBMS. Our extension reduces the distance between the conceptual design and the physical implementation of spatial databases. AST-PostGIS offers advanced representations for geo-object and geo-field geometries, along with procedures to assert the consistency of spatial relationships during data insertions, updates and deletes. Special procedures supported by the extension can be used to check the consistency of database before enforcing spatial integrity constraints for the first time, recording inconsistencies in a special log table.

AST-PostGIS was written in PL/pgSQL and is currently available for PostgreSQL/PostGIS. However, it can be adapted with relative simplicity to any other extensible spatial RDBMS, since AST-PostGIS mechanisms use the primitive types standardized by the OGC. The advanced spatial data types can be handled as any RDBMS type, i.e., they can be used as column definitions of tables, as variables in scripts or as arguments of functions or stored procedures. Applying the spatial integrity constraints requires complex triggers, but this complexity was necessary to overcome the limitations of the RDBMS for supporting spatial relationships. A design tool is capable of generating complete DDL scripts, including the necessary triggers, from OMT-G class diagrams.

The successful implementation of AST-PostGIS shows that SQL-based RDBMSs can evolve in order to natively support spatial data, along with the necessary functions, integrity constraints and tools, without resorting to extensions.

Our PostGIS extension is functional and simple to use. To demonstrate its operation in this work, we presented a compact but comprehensive example of an urban cadastral system. In this test, we first presented the conceptual schema fragment that

models a geographic area of a municipality. Then we showed the physical implementation of this schema by using the AST-PostGIS features. Finally, we took three cases of spatial integrity violations that might occur, and showed how AST-PostGIS responds to them.

Future work includes creating benchmarks to evaluate how AST-PostGIS and its advanced spatial data types and functions perform with larger database schemas. Spatial data demands more complex data structures and have a potentially slower performance when compared to traditional data. Therefore, it is important to evaluate the performance of each procedure of AST-PostGIS individually. We also intend to use AST-PostGIS consistency check functions to search for inconsistencies in production-grade spatial datasets. Furthermore, we also plan to adapt AST-PostGIS for other spatial RDBMSs, including proprietary systems, like Oracle Spatial.

Bibliography

- Abiteboul, S. and Hull, R. (1987). IFO: a formal semantic database model. *TODS: ACM Transactions on Database Systems*, 12(4):525–565.
- Abrantes, G. and Carapuça, R. (1994). Explicit representation of data that depend on topological relationships and control over data consistency. In Harts, J. J., Ottens, H. F., and Scholten, H. J., editors, *Proceedings of the 5th European Conference and Exhibition on Geographical Information Systems, EGIS/MARI '94*, pages 869–877, Paris, France. Utrecht : EGIS Foundation.
- Adler, D. W. (2001). DB2 Spatial Extender – spatial data within the RDBMS. In Apers, P. M. G., Atzeni, P., Ceri, S., Paraboschi, S., Ramamohanarao, K., and Snodgrass, R. T., editors, *Proceedings of the 27th International Conference on Very Large Data Bases, VLDB '01*, pages 687–690, Roma, Italy. Morgan Kaufmann Publishers Inc.
- Aitchison, A. (2009). *Beginning Spatial with SQL Server 2008*. Apress.
- Bédard, Y., Caron, C., Maamar, Z., Moulin, B., and Vallière, D. (1996). Adapting data models for the design of spatio-temporal databases. *Computers, Environment and Urban Systems*, 20(1):19–41.
- Booch, G., Rumbaugh, J., and Jacobson, I. (2005). *The Unified Modeling Language User Guide*. Addison-Wesley Professional, 2nd edition.
- Borges, K. A. V. (1997). *Modelagem de dados geográficos – uma extensão do modelo OMT para aplicações geográficas*. Master thesis, Fundação João Pinheiro, Belo Horizonte, Brazil.
- Borges, K. A. V., Davis Jr., C. A., and Laender, A. H. F. (2001). OMT-G: an object-oriented data model for geographic applications. *GeoInformatica: An International Journal on Advances of Computer Science for Geographic Information Systems*, 5(3):221–260.

- Borges, K. A. V., Davis Jr., C. A., and Laender, A. H. F. (2002). Integrity constraints in spatial databases. In Doorn, J. H. and Rivero, L. C., editors, *Database Integrity: Challenges and Solutions*, chapter 5, pages 144–171. Idea Group.
- Borges, K. A. V., Davis Jr., C. A., and Laender, A. H. F. (2005). Modelagem conceitual de dados geográficos. In Casanova, M. A., Câmara, G., Davis Jr., C. A., Vinhas, L., and Queiroz, G. R., editors, *Banco de dados geográficos*, chapter 3, pages 93–146. MundoGEO.
- Boundless Suite User Manual (2016). Modelling object interactions. [Diagram]. Retrieved from: <https://connect.boundlessgeo.com/docs/suite/enterprise/latest/dataadmin/pgAdvanced/de9im.html>.
- Chen, P. P. (1976). The entity-relationship model – toward a unified view of data. *TODS: ACM Transactions on Database Systems*, 1(1):9–36.
- Chrisman, N. R. (1991). The error component in spatial data. In Maguire, D. J., Goodchild, M. F., and Rhind, D. W., editors, *Geographical Information Systems: Principles and Applications*, volume 1, chapter 12, pages 165–174. Longman Scientific and Technical.
- Clementini, E. and Di Felice, P. (2000). Spatial operators. *ACM SIGMOD Record*, 29(3):31–38.
- Clementini, E., Felice, P. D., and Oosterom, P. V. (1993). A small set of formal topological relationships suitable for end-user interaction. In Abel, D. and Ooi, B. C., editors, *Advances in Spatial Databases: 3rd International Symposium, SSD '93*, volume 692 of *Lecture Notes in Computer Science*, pages 277–295, Singapore. Springer.
- Cockcroft, S. (1997). A taxonomy of spatial data integrity constraints. *GeoInformatica: An International Journal on Advances of Computer Science for Geographic Information Systems*, 1(4):327–343.
- CUBRID Blog (2016). Spatial RDBMS architecture. [Diagram]. Retrieved from: <http://cubrid.org/blog/20-minutes-to-understanding-spatial-database/>.
- Câmara, G., Casanova, M. A., Hemerly, A. S., Magalhães, G. C., and Medeiros, C. M. B. (1996). *Anatomia de sistemas de informação geográfica*. UNICAMP.
- Davis Jr., C. A., Borges, K. A. V., and Laender, A. H. F. (2001). Restrições de integridade em bancos de dados geográficos. In *Proceedings of the 3rd Brazilian*

- Workshop on GeoInformatics, GEOINFO 2001*, pages 63–70, Rio de Janeiro, Brazil. Instituto Nacional de Pesquisas Espaciais (INPE).
- Davis Jr., C. A., Borges, K. A. V., and Laender, A. H. F. (2005). Deriving spatial integrity constraints from geographic application schemas. In Rivero, L. C., Doorn, J. H., and Ferragine, V. E., editors, *Encyclopedia of Database Technologies and Applications*, pages 176–183. Idea Group.
- Douglas, K. and Douglas, S. (2005). *PostgreSQL: a comprehensive guide to building, programming, and administering PostgreSQL databases*. Sams Publishing, 2nd edition.
- Egenhofer, M. J. (1989). A formal definition of binary topological relationships. In Litwin, W. and Schek, H.-J., editors, *Proceedings of the 3rd International Conference on Foundations of Data Organization and Algorithms, FODO 1989*, volume 367 of *Lecture Notes in Computer Science*, pages 457–472, Paris, France. Springer.
- Egenhofer, M. J. and Franzosa, R. D. (1991). Point-set topological spatial relations. *IJGIS: International Journal of Geographical Information Systems*, 5(2):161–174.
- Egenhofer, M. J. and Herring, J. (1990). A mathematical framework for the definition of topological relationships. In Brassel, K. E. and Kishimoto, H., editors, *Proceedings of the 4th International Symposium on Spatial Data Handling*, pages 803–813, Zurich, Switzerland. International Geographical Union, Commission on Geographical Data Sensing and Processing, Department of Geography, Ohio State University.
- Egenhofer, M. J. and Herring, J. (1994). Categorizing binary topological relations between regions, lines, and points in geographic databases. In Egenhofer, M. J., Mark, D. M., and Herring, J., editors, *The 9-Intersection: formalism and its use of natural-language spatial predicates (94-1)*, NCGIA Technical Papers. UC Santa Barbara: National Center for Geographic Information and Analysis.
- Elmasri, R. and Navathe, S. B. (2015). *Fundamentals of Database Systems*. Pearson, 7th edition.
- Fang, Y., Friedman, M., Nair, G., Rys, M., and Schmid, A. (2008). Spatial indexing in Microsoft SQL Server 2008. In Wang, J. T., editor, *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pages 1207–1216, New York, USA. ACM.

- Fatto, V. D., Deufemia, V., Paolino, L., and Tumiati, S. (2015). Wispy: A tool for visual specification and verification of spatial integrity constraints. *VLSS: Journal of Visual Languages and Sentient Systems*, 1:39–48.
- Frank, A. U. and Mark, D. M. (1991). Language issues for geographical information systems. In Maguire, D. J., Goodchild, M. F., and Rhind, D. W., editors, *Geographical Information Systems: Principles and Applications*, volume 1, chapter 11, pages 147–163. Longman Scientific and Technical.
- Güting, R. H. (1994). An introduction to spatial database systems. *The VLDB Journal – The International Journal on Very Large Data Bases*, 3(4):357–399.
- Han, J., Haihong, E., Le, G., and Du, J. (2011). Survey on NoSQL database. In Hu, B., Zu, Q., van Greunen, D., Hu, B., Ma, H., and Ning, Y., editors, *Proceedings of the 6th International Conference on Pervasive Computing and Applications*, ICPCA 2011, pages 363–366, Port Elizabeth, South Africa. IEEE.
- Hoop, S. and Van Oosterom, P. (1992). Storage and manipulation of topology in Postgres. In Harts, J., Ottens, H. F., and Scholten, H. J., editors, *Proceedings of the 3rd European Conference and Exhibition on Geographical Information Systems, EGIS '92*, pages 1324–1336, Munich, Germany. EGIS Foundation.
- Hora, A. C., Davis Jr., C. A., and Moro, M. M. (2010). Generating XML/GML schemas from geographic conceptual schemas. In Laender, A. H. F. and Lakshmanan, L. V. S., editors, *AMW 2010, Proceedings of the 4th Alberto Mendelzon International Workshop on Foundations of Data Management*, volume 619 of *CEUR Workshop Proceedings*, Buenos Aires, Argentina. CEUR-WS.org.
- Hora, A. C., Davis Jr., C. A., and Moro, M. M. (2011). Mapping network relationships from spatial database schemas to GML documents. *JIMD: Journal of Information and Data Management*, 2(1):67–74.
- ISO/IEC 13249-3 (2016). Information technology – Database languages – SQL multimedia and application packages – Part 3: Spatial. International Standard ISO/IEC 13249-3:2016(E), International Organization for Standardization, Geneva, Switzerland.
- Kemp, K. K. (1992). *Environmental modeling with GIS: a strategy for dealing with spatial continuity*. PhD thesis, University of California, Santa Barbara, USA.

- Kosters, G., Pagel, B., and Six, H. (1997). GIS-application development with GeoOOA. *IJGIS: International Journal of Geographical Information Science*, 11(4):307–335.
- Laurini, R. and Thompson, D. (1992). *Fundamentals of Spatial Information Systems*. Number 37 in The A.P.I.C. Series. Academic Press, 7th edition.
- Lisboa Filho, J., Costa, A. C., and Iochpe, C. (1999). Projeto de banco de dados geográficos: mapeando esquemas GeoFrame para o SIG Spring. In Paiva, J. A. d. C., editor, *Proceedings of the 1rd Brazilian Workshop on GeoInformatics, GEOINFO 1999*, Campinas, Brazil. Instituto Nacional de Pesquisas Espaciais (INPE).
- Lisboa Filho, J. and Iochpe, C. (1999). Specifying analysis patterns for geographic databases on the basis of a conceptual framework. In Medeiros, C. B., editor, *Proceedings of the 7th ACM International Symposium on Advances in Geographic Information Systems, GIS '99*, pages 7–13, Kansas City, USA. ACM.
- Lisboa Filho, J., Rodrigues Jr., M. F., Daltio, J., and de Freitas Sodré, V. (2004a). ArgoCASEGEO – an open source CASE tool for geographic information systems modeling using the UML-GeoFrame model. In *Proceedings of the 7th International Conference on Information Systems Implementation and Modeling, ISIM '04*, pages 29–36, Ostrava, Czech Republic.
- Lisboa Filho, J., Sodré, V. F., Daltio, J., Rodrigues Jr., M. F., and Vilela, V. M. (2004b). A CASE tool for geographic database design supporting analysis patterns. In Wang, S., Tanaka, K., Zhou, S., Ling, T.-W., Guan, J., Yang, D.-q., Grandi, F., Mangina, E. E., Song, I.-Y., and Mayr, H. C., editors, *Proceedings of Conceptual Modeling for Advanced Application Domains, ER 2004 Workshops CoMoGIS, COMWIM, ECDM, CoMoA, DGOV, and ECOMO*, volume 3289 of *Lecture Notes in Computer Science*, pages 43–54, Shanghai, China. Springer.
- Lizardo, L. E. O. and Davis Jr., C. A. (2014). OMT-G Designer: a Web tool for modeling geographic databases in OMT-G. In Indulska, M. and Purao, S., editors, *Advances in Conceptual Modeling: 33rd International Conference on Conceptual Modeling, ER 2014*, volume 8823 of *Lecture Notes in Computer Science*, pages 228–233, Atlanta, USA. Springer.
- Lizardo, L. E. O., Moro, M. M., and Davis Jr., C. A. (2014). GeoNoSQL: Banco de dados geoespacial em NoSQL. In Miranda, E. M., editor, *Proceedings of The Computer on the Beach*, pages 303–309, Florianópolis, Brazil. Universidade do Vale do Itajaí - UNIVALI, Centro de Ciências Tecnológicas da Terra e do Mar - CTTMar.

- Melton, J. and Eisenberg, A. (2001). SQL multimedia and application packages (SQL/MM). *ACM SIGMOD Record*, 30(4):97–102.
- Melton, J. and Simon, A. (1993). *Understanding the New SQL: A Complete Guide*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers.
- Momjian, B. (2001). *PostgreSQL: Introduction and Concepts*. Addison-Wesley.
- Obe, R. O. and Hsu, L. S. (2015). *PostGIS in action*. Manning Publications, 2nd edition.
- OGC 06-103r4 (2011). OpenGIS Implementation Standard for Geographic information - Simple feature access – Part 1: Common architecture. OpenGIS Implementation Standard OGC 06-103r4, Open Geospatial Consortium Inc.
- OGC 06-104r4 (2010). OpenGIS Implementation Standard for Geographic information - Simple feature access – Part 2: SQL option. OpenGIS Implementation Standard OGC 06-104r4, Open Geospatial Consortium Inc.
- Oliveira, J. L., Pires, F., and Medeiros, C. B. (1997). An environment for modeling and design of geographic applications. *GeoInformatica: An International Journal on Advances of Computer Science for Geographic Information Systems*, 1(1):29–58.
- Oracle (2017). Spatial and Graph Analytics with Oracle Database 12c Release 2. Technical white paper, Oracle Corporation.
- Piórkowski, A. (2011). MySQL Spatial and PostGIS – implementations of spatial data standards. *EJPAU: Electronic Journal of Polish Agricultural Universities*, 14(1):1–8.
- Pullar, D. V. and Egenhofer, M. J. (1988). Toward formal definitions of topological relations among spatial objects. In *Proceedings of the 3rd International Symposium on Spatial Data Handling*, pages 225–241, Sydney, Australia. International Geographical Union, Commission on Geographical Data Sensing and Processing, Department of Geography, Ohio State University.
- Queiroz, G. R. d., Monteiro, A. M. V., and Câmara, G. (2013). Bancos de dados geográficos e sistemas NoSQL: onde estamos e para onde vamos. *Revista Brasileira de Cartografia*, 65(3):479–492.
- Ramsey, P. et al. (2005). *PostGIS 2.0 Manual*. Refrations Research Inc.

- Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W. E., et al. (1991). *Object-Oriented Modeling and Design*. Prentice Hall.
- Santos, P. O., Moro, M. M., and Davis Jr., C. A. (2015). Comparative performance evaluation of relational and NoSQL databases for spatial and mobile applications. In Chen, Q., Hameurlain, A., Toumani, F., Wagner, R., and Decker, H., editors, *Proceedings of the 26th International Conference on Database and Expert Systems Applications, DEXA 2015*, volume 9261 of *Lecture Notes in Computer Science*, pages 186–200, Valencia, Spain. Springer.
- Seufitelli, D. B., Moro, M. M., and Davis Jr., C. A. (2015). Desafios no mapeamento de esquemas conceituais geográficos para esquemas físicos híbridos SQL/NoSQL. In Körting, T. S. and Fileto, R., editors, *Proceedings of the 16th Brazilian Workshop on GeoInformatics, GEOINFO 2015*, pages 119–124, Campos do Jordão, Brazil. Instituto Nacional de Pesquisas Espaciais (INPE).
- Shekhar, S., Coyle, M., Goyal, B., Liu, D.-R., and Sarkar, S. (1997). Data models in geographic information systems. *Communications of the ACM*, 40(4):103–111.
- Shukla, D., Shivnani, C., and Shah, D. (2016). Comparing Oracle Spatial and Postgres PostGIS. *IJCSC: International Journal of Computer Science and Communication*, *IJCSC*, 7(2):95–100.
- Souza, A. C. V., Amaral, H. R., and Lizardo, L. E. O. (2011). PostgreSQL: uma alternativa para sistemas gerenciadores de banco de dados de código aberto. In Matte, A. C. F., Araujo, A., Marinho, A. R. P., and Quadros, G. B. F., editors, *Proceedings of the Congresso Nacional Universidade, EAD e Software Livre – UEADSL 2011*, volume 2, Belo Horizonte, Brazil.
- Stolze, K. (2003). SQL/MM Spatial – the standard to manage spatial data in a relational database system. In Weikum, G., Schöning, H., and Rahm, E., editors, *Tagungsband der 10. BTW-Konferenz Datenbanksysteme für Business, Technologie und Web, BTW 2003*, volume 26 of *LNI*, pages 247–264, Leipzig, Germany. GI.
- Strobl, C. (2008). Dimensionally Extended Nine-Intersection Model (DE-9IM). In Shekhar, S. and Xiong, H., editors, *Encyclopedia of GIS*, pages 240–245, Boston, USA. Springer.
- Surve, R. B. and Kathane, B. Y. (2014). Disparity of spatial and non spatial data. *IJAFRC: International Journal of Advance Foundation and Research in Computer*, 1(8):56–62.

- Widenius, M. and Axmark, D. (2002). *MySQL Reference Manual: Documentation from the Source*. O'Reilly Media.
- Worboys, M. F., Hearnshaw, H. M., and Maguire, D. J. (1990). Object-oriented data modelling for spatial databases. *IJGIS: International Journal of Geographical Information Systems*, 4(4):369–383.
- Zhou, Z., Zhou, B., Li, W., Griglak, B., Caiseda, C., and Huang, Q. (2009). Evaluating query performance on object-relational spatial databases. In *Proceedings of the 2nd IEEE International Conference on Computer Science and Information Technology, ICCSIT 2009*, pages 489–492, Beijing, China. IEEE.

Appendix A

SQL schema for the urban cadastral database

```
-- Table City
CREATE TABLE City (
    id integer PRIMARY KEY,
    name varchar(30),
    geom_point ast_point,
    geom_boundary ast_polygon
);

-- Table Relief
CREATE TABLE Relief (
    altitude integer,
    geom ast_isoline
);

-- Table Block
CREATE TABLE Block (
    zoning_type varchar(10),
    geom ast_polygon
);

-- Table Parcel
CREATE TABLE Parcel (
    id integer PRIMARY KEY,
    geom ast_polygon
);
```

```
-- Table Residential
CREATE TABLE Residential (
    buiding_id integer PRIMARY KEY,
    buiding_number integer,
    geom ast_polygon
);

-- Table Commerce
CREATE TABLE Commerce (
    buiding_id integer PRIMARY KEY,
    buiding_number integer,
    name varchar(50),
    activity varchar(50),
    geom ast_polygon
);

-- Table Industry
CREATE TABLE Industry (
    buiding_id integer PRIMARY KEY,
    buiding_number integer,
    name varchar(50),
    production_type varchar(50),
    geom ast_polygon
);

-- Table Nature_Reserve
CREATE TABLE Nature_Reserve (
    name varchar(50) PRIMARY KEY,
    geom ast_polygon
);

-- Table Neighborhood
CREATE TABLE Neighborhood (
    id integer PRIMARY KEY,
    name varchar(50),
    geom ast_planarsubdivision
);
```

```
-- Conventional table Thoroughfare
CREATE TABLE Thoroughfare (
  name varchar(50) PRIMARY KEY,
  speed_limit integer
);

-- Table Address
CREATE TABLE Address (
  number integer,
  thoroughfare varchar(50)
    REFERENCES Thoroughfare(name),
  geom ast_point
);

-- Table Street_Segment
CREATE TABLE Street_Segment (
  pavement varchar(10),
  thoroughfare varchar(50)
    REFERENCES Thoroughfare(name),
  geom ast_biline
);

-- Table Crossing
CREATE TABLE Crossing (
  geom ast_node
);
```

Listing A.1: Tables definition

```
-- City Point
CREATE INDEX SIDX_City_Point
  ON City
  USING GIST (geom_point);
-- City Boundary
CREATE INDEX SIDX_City_Boundary
  ON City
  USING GIST (geom_boundary);
-- Relief
CREATE INDEX SIDX_Relief
  ON Relief
  USING GIST (geom);
-- Block
CREATE INDEX SIDX_Block
  ON Block
  USING GIST (geom);
-- Parcel
CREATE INDEX SIDX_Parcel
  ON Parcel
  USING GIST (geom);
-- Residential
CREATE INDEX SIDX_Residential
  ON Residential
  USING GIST (geom);
-- Commerce
CREATE INDEX SIDX_Commerce
  ON Commerce
  USING GIST (geom);
-- Industry
CREATE INDEX SIDX_Industry
  ON Industry
  USING GIST (geom);
-- Nature_Reserve
CREATE INDEX SIDX_Nature_Reserve
  ON Nature_Reserve
  USING GIST (geom);
```



```
-- Neighborhood
CREATE INDEX SIDX_Neighborhood
  ON Neighborhood
  USING GIST (geom);
-- Address
CREATE INDEX SIDX_Address
  ON Address
  USING GIST (geom);
-- Street_Segment
CREATE INDEX SIDX_Street_Segment
  ON Street_Segment
  USING GIST (geom);
-- Crossing
CREATE INDEX SIDX_Crossing
  ON Crossing
  USING GIST (geom);
```

Listing A.2: Indexes definition

```
-- Spatial aggregation between Block and Parcel
CREATE TRIGGER aggregation_block_parcel
  AFTER INSERT OR UPDATE OR DELETE ON Parcel
  FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_aggregation('Parcel', 'geom', 'Block', 'geom');

-- Spatial aggregation between City and Neighborhood
CREATE TRIGGER aggregation_boundary_neighborhood
  AFTER INSERT OR UPDATE OR DELETE ON Neighborhood
  FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_aggregation('Neighborhood', 'geom', 'City',
      'geom_boundary');

-- Spatial relationship between Residential and Parcel
CREATE TRIGGER spatial_residential_parcel
  AFTER INSERT OR UPDATE ON Residential
  FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_spatialrelationship('Residential', 'geom',
      'Parcel', 'geom', 'within');

-- Spatial relationship between Commerce and Parcel
CREATE TRIGGER spatial_commerce_parcel
  AFTER INSERT OR UPDATE ON Commerce
  FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_spatialrelationship('Commerce', 'geom',
      'Parcel', 'geom', 'within');

-- Spatial relationship between Industry and Parcel
CREATE TRIGGER spatial_industry_parcel
  AFTER INSERT OR UPDATE ON Industry
  FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_spatialrelationship('Industry', 'geom',
      'Parcel', 'geom', 'within');
```

```

-- Spatial relationship between Block and City
CREATE TRIGGER spatial_block_city
AFTER INSERT OR UPDATE ON Block
FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_spatialrelationship('Block', 'geom',
        'City', 'geom_boundary', 'within');

-- Spatial relationship between Parcel and Address
CREATE TRIGGER spatial_parcel_address
AFTER INSERT OR UPDATE ON Parcel
FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_spatialrelationship('Parcel', 'geom',
        'Address', 'geom', 'contains');

-- Spatial relationship between Nature and Industry
CREATE TRIGGER spatial_nature_distant
AFTER INSERT OR UPDATE ON Nature_Reserve
FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_spatialrelationship('Nature_Reserve',
        'geom', 'Industry', 'geom', 'distant', '800');

-- Arc-Node network between Street_Segment and Crossing
CREATE TRIGGER network_street_crossing
AFTER INSERT OR UPDATE ON Street_Segment
FOR EACH STATEMENT EXECUTE PROCEDURE
    ast_arcnodenetwork('Street_Segment', 'geom',
        'Crossing', 'geom');

```

Listing A.3: Triggers definition