A PostGIS extension to support advanced spatial data types and integrity constraints

Conference Paper · November 2017

DOI: 10.1145/3139958.3140020

2 authors:

Luis Eduardo Oliveira Lizardo
IBM Research & Development, Germany
2 PUBLICATIONS **6** CITATIONS
SEE PROFILE

Clodoveu Augusto Davis Jr.
Federal University of Minas Gerais
186 PUBLICATIONS **2,394** CITATIONS
SEE PROFILE

A PostGIS extension to support advanced spatial data types and integrity constraints

Luís Eduardo Oliveira Lizardo, Clodoveu Augusto Davis Jr.
{lizardo,clodoveu}@dcc.ufmg.br
Department of Computer Science
Universidade Federal de Minas Gerais
Belo Horizonte, Brazil

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 directly 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. This paper 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, mapping its conceptual schema to the physical implementation in extended SQL.

CCS CONCEPTS

• **Information systems → Geographic information systems**; *Integrity checking*;

KEYWORDS

Spatial databases, Spatial data modeling, Spatial integrity Constraints, OMT-G, Geographic Information Systems

ACM Reference format:
Luís Eduardo Oliveira Lizardo, Clodoveu Augusto Davis Jr.. 2017. A PostGIS extension to support advanced spatial data types and integrity constraints. In *Proceedings of SIGSPATIAL'17, Los Angeles Area, CA, USA, November 7–10, 2017,* 10 pages.
https://doi.org/10.1145/3139958.3140020

1 INTRODUCTION

OpenGIS 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) [28, 29], 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 geometric constraints, such as the identification of non-simple linestrings, and establishes the Dimensionally-Extended Nine-Intersection Model (DE-9IM) [11] as the basis for establishing relationships and enforcing topological constraints. The SQL/MM standard (Part Three - Spatial) [18, 25], derived from OpenGIS, provides more functions, more dimensions for objects and includes considerations on spatial representations, defining which functions must be used to compare, transform and process spatial data [35].

Since 1999, these standards were progressively adopted by popular DBMSs. For instance, Oracle Spatial[1] complies with OpenGIS SFS and supports SQL/MM types and operators [31]. MySQL Spatial[2] extension implements part of the OpenGIS standard and only the 2D representations, without reference sets [32, 36]. Microsoft's SQL Server Spatial Storage[3] conforms to the OpenGIS SFS, but not to its geography data types [4, 14]. IBM DB2 Spatial Extender[4] implements types and functions defined by both specifications [3]. PostGIS[5], the open source spatial extension for PostgreSQL[6], conforms to both standards almost completely [27, 32].

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 example, the mapping of planar subdivisions, sets of polygons in

[1]https://www.oracle.com/database/spatial/
[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/

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

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 must 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. Ideally, the DDL of a spatial DBMS should include primitives for checking and enforcing spatial integrity constraints, but that is not in the SFS. Implementing and re-implementing such generic constraint verification code using more basic functions, 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.

In this paper we present AST-PostGIS, an extension that implements advanced spatial data types and spatial integrity constraints over PostGIS. Our objective is to reduce the gap between the conceptual and physical schemas in spatial database design, and show that richer geographic types and spatial integrity constraints could be included in spatial DBMSs. AST-PostGIS also provides mechanisms to identify integrity constraint violations on spatial databases upon initial enforcement, and to trigger procedures that constrain the insertion or deletion of inconsistent spatial data in the RDMBS, following conceptual design semantics. Naturally, the ideas and definitions presented here and implemented for PostGIS/PostgreSQL can be adapted for any other extensible RDBMS with support for spatial data [26]. AST-PostGIS is open-source[7] and cross-platform, easy to install and enable in each database schema. The spatial data types and integrity constraints are based on those defined by OMT-G, an object oriented data model for the design of geographic applications and geographic database systems [7].

The remainder of this paper is organized as follows. Related work is reviewed in Section 2. Section 3 presents the OMT-G data model and gives a brief overview of its primitives for conceptual modeling of geographic data. AST-PostGIS is introduced in Section 4. Section 5 illustrates the use of the AST-PostGIS with the implementation of a small urban geographic database schema. Finally, Section 6 discusses the conclusions and future work.

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

2 RELATED WORK

The first data models for spatial applications were guided by existing GIS internal structures. 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 [9]. Conventional semantic and object-oriented data models, like ER [10], OMT [33], IFO [1] and UML [6] 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 [30]. 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 [16] 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 capturing the semantics of geographic data, offering mechanisms of higher abstraction and implementation independence. Concepts like geometry and topology are fundamentally important 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 fault lines and river networks can be discretized. Depending on the level of detail considered, some real-world entities can even belong to both categories [19].

Various spatial data models have been proposed in the literature. For example, Worboys et al. [37] proposed EXT.IFO, an IFO Model [1] 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 [2] extended the OMT Data Model [6] and created OMT EXT, with primitives for modeling topological relationships, namely *partition*, *covering* and *disconnected class*, the latter being a concept associated with the subclasses derived from *partitions* and *coverings*.

GISER [34] and GMOD [30] do not define specific modeling primitives and cannot be considered proper data models [7]. They only provide modeling patterns to be followed by geographic application designers. GISER extends the ER Model [10] and integrates field-based and object-based models of geographic data by using the *discretized-by* relationship between feature fields and coverage entities. It has predefined entities and relationships that represent fields, objects, 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 can also model the geometry of spatial entities, as well as temporal aspects. GMOD introduces new relationships between entities, e.g. (*causal* and *version*).

Bédard et al. [5] introduced MODUL-R with a fixed set of geometric types that use pictograms to represent the geometric shapes of entities. The combination of these pictograms can represent multiple views of the same entity. MODUL-R, on the other hand, does not distinguish between fields and objects and does not have primitives to represent topological connectivity, or spatial aggregation. GeoOOA [20] is a model that 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 a spatial representation are distinguished by this model, although it lacks the support for spatial integrity constraints and does not represent fields properly.

Lisboa Filho and Iochpe [21] proposed GeoFrame, a conceptual geographic data model with a hierarchical class structure. The hierarchy is subdivided in three 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 defines associated themes (*Theme* class), such as limits of the urban area, hydrography, relief, and so on. 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. It can be represented by conventional data or geographic phenomena, the latter being specialized in meta-classes for fields and objects. The *SpatialObject* class generalizes spatial representation classes observed in the 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. Other works introduced tools to support the model [22, 23].

To the best of our knowledge, the most complete spatial data model is OMT-G [7], 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 its diagrams tend to be more compact than others, because of the higher semantic content of its primitives. OMT-G motivated many initiatives, including Wispy [15], a uDig[8] extension that permits verifying and visually specifying complex spatial integrity constraints.

In an earlier work, Borges et al. [9] 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. Hora et al. [17] later implemented an OMT-G mapping tool to generate Oracle PL/SQL schemas[9], that includes triggers, procedures and XML schemas[10]. Lizardo and Davis Jr. [24] present OMT-G Designer[11], 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. The OMT-G model is described in more detail in Section 3.

[8]uDig: http://udig.refractions.net/
[9]OMTG2SQL: https://github.com/lab-csx-ufmg/omtg2sql
[10]OMTG2GML: https://github.com/lab-csx-ufmg/omtg2gml
[11]OMT-G Designer: http://aqui.io/omtg/

3 THE OMT-G MODEL

3.1 Overview

OMT-G (*Object Modeling Technique for Geographic Applications*) [7] is an object-oriented data model for the design of geographic applications and geographic database systems. It provides 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. 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. Two types of class primitives are specified by the OMT-G model: *conventional* and *georeferenced*. The georeferenced class notation has a top left-hand rectangle that indicates the geometry of the representation, whereas the notation used for conventional classes is similar to the notation used in UML [6], as shown in Figure 1.

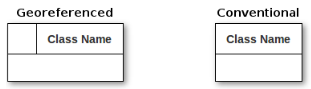


Figure 1: Simplified graphical notation of a class in OMT-G.

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 include a geographic representation alternative, which specializes into discrete, associated with real world elements (*geo-objects*), or continuously distributed over the space (*geo-fields*). Geo-objects are represented with points, lines, polygons or network elements (nodes, unidirectional and bidirectional arcs). Geo-fields can be represented as isolines, tesselation, planar subdivision, sampling or triangular irregular network (TIN). Figures 2 and 3 show, respectively, examples of geo-object and geo-field classes.

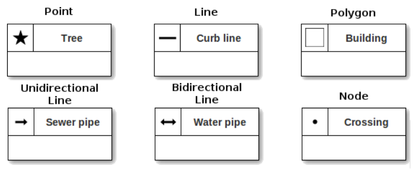


Figure 2: Geo-object classes

Relationships can also be conventional, i.e., simple associations, such as in UML relationships, or georeferenced. The latter include

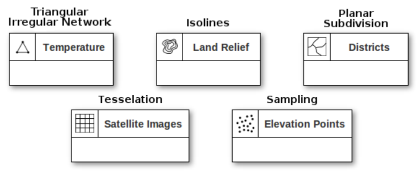
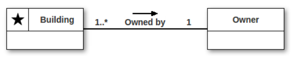
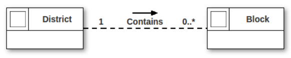


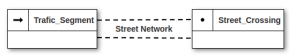
Figure 3: Geo-field classes



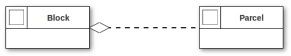
(a) Simple association



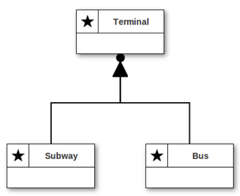
(b) Spatial relationship



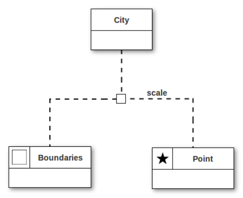
(c) Arc-node network relationship



(d) Aggregation



(e) Generalization



(f) Conceptual Generalization

Figure 4: Relationships and generalizations.

topological relationships (e.g. touch, in, cross, overlap, disjoint etc), *arc-node networks* and *spatial aggregations* (i.e. "whole-part" aggregations). *Generalizations* and *specializations* can be disjoint/overlapping or total/partial, and require that the participating classes have the same type of representation. The *conceptual generalization* primitive allows modeling objects with multiple geographic representations, which may vary according to the scale or to the geometric shape. Figure 4 shows OMT-G notations for relationships.

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 [8, 12, 13]. 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. Other constraints apply to the geometric representation of geo-objects (i.e., constraints related to the consistency of lines and polygons).

The integrity rules are presented in detail next, grouped into categories.

3.2.1 *Geo-field constraints.*

Five spatial integrity rules derive from the semantics involved in the concept of geo-fields and from the specific definition of four geo-field representation alternatives.

(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. A set of n *isolines* L represents F if (1) every component isoline L_i is a polygonal line (see rule **R6**), and (2) the value of F at every point P such that $P \in L_i$, $0 \le i \le n$, is constant.

(R3) Tesselation: Let F be a geo-field. Let $C = \{c_0, c_1, c_2, \ldots, c_n\}$ be a set of regularly-shaped cells covering F. C is a tesselation 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 $A = \{A_0, A_1, A_2, \ldots, A_n\}$ be a set of polygons and F be a geo-field. Such that $A_i \subset F$ for all i such that $0 \le i \le n$. 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 $T = \{T_0, T_1, T_2, \ldots, T_n\}$ be a set of triangles such that $T_i \subset F$ for all i such that $0 \le i \le n$. 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 lines and polygons lead to some integrity constraints, which affect all linear or areal representations, such as network arcs or polygonal components of planar subdivisions. Rules R6 to R8 regard lines, simple polygons and polygonal regions.

(R6) Line: Let $v_0, v_1, \ldots, v_n$ be $n+1$ points in the plane. Let $a_0 = \overline{v_0 v_1}$, $a_1 = \overline{v_1 v_2}$, ..., $a_{n-1} = \overline{v_{n-1} v_n}$ be n segments, connecting the points. These segments form an *simple 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 $j \neq i + 1$). By this definition, every line must be a *simple polygonal line.*

(R7) Simple Polygon: A *simple polygon* is a simple polygonal line in which the last vertex coincides with the first vertex, i.e., $v_0 = v_n$, that is, the polygon is closed.

(R8) Polygonal Region: Let $R = \{P_0, P_1, ..., P_{n-1}\}$ 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 arranged in a counterclockwise sequence, and (3) P_0 *contains* P_i for all $P_i \neq P_0$ in which the vertices are arranged clockwise.

3.2.3 *Network relationship constraints.*

Network relationships involve arc and node objects that are connected with each other. OMT-G considers networks with no nodes, in which connections occur at the geometric endpoints of arcs. The connectivity rules are defined in rules R9 and R10.

(R9) Arc-node network: Let $G = \{N, A\}$ be a network structure composed of a set of nodes $N = \left\{n_0, n_1, \ldots, n_p\right\}$ and a set of arcs $A = \left\{a_0, a_1, \ldots, a_q\right\}$. Members of N are related to members of A according to the following constraints: for every node $n_i \in N$ there must be at least one arc $a_k \in A$, and for every arc $a_k \in A$ there must be exactly two nodes n_i, $n_j \in N$.

(R10) Arc-arc network: Arc-arc network:] Let $G = \{A\}$ be a network structure composed of a set of arcs $A = \{a_0, a_1, \ldots, a_q\}$. Then 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. Spatial aggregation is a special form of association between objects, where one of them is considered to be geometrically assembled from others. 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 $P = \{P_0, P_1, \ldots, P_n\}$ 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) $\left(W \cap \bigcup_{i=0}^{n} P_i\right) = W$, and (3) $((P_i \ touch \ P_j) \vee (P_i \ disjoint \ P_j)) =$ 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. Most relationships are derived from comparing the geometries of objects. Others need to be specified by the user, in order to allow the system to actually establish the connection between objects. OMT-G considers a set of five basic spatial relationships between georeferenced classes, from which all others can be derived [11, 13]: crosses, disjoint, overlaps, touches and within. The OGC, however, later established the dimensionally extended 9-intersection matrix (DE-9IM) as a basis for the implementation of spatial relationship functions in spatial DBMSs.

In fact, most spatial relationships denoted in OMT-G class diagrams actually indicate spatial integrity constraints, rather than the need for materialized connections, such as primary key/foreign key pairs in relational tables. Spatial relationships can be indicated in OMT-G diagrams by their conventional name, and established as constraints based on DE-9IM [11] functions whenever the relationship type and its cardinality indicate a constraint. We refrain from presenting in detail each spatial relationship constraint due to lack of space, and will henceforth refer generically to such rules as **RX**.

3.3 Mapping OMT-G to physical schemas

Geometric types in conceptual schemas are accompanied by an expected behavior, captured with a set of spatial integrity constraints [8]. 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*, *tesselation*, *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, as defined by OGC standards. When mapping a spatial data schema from the conceptual to the physical level, we are forced to use these simple geometric representations that are available in the spatial RDBMS. This process, if executed directly, implies in loss of semantics, since the only topological constraints implemented in the spatial RDBMS DDL syntax are simple value checks. 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. Implementing such code is not trivial, and demands advanced knowledge of the resources offered by the RDBMS. However, given the set of spatial integrity constraints that can be extracted directly from OMT-G class diagrams, a generic implementation for each type of constraint can be achieved.

Table 1: Geometric types: OMT-G and OGC

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
TIN	Point and Polygon
Tesselation	–

Table 1 shows how the OMT-G primitives are mapped to corresponding (albeit semantically poorer) OGC representations. For instance, the representations *line*, *unidirectional arc*, *bidirectional arc* and *isolines* are all mapped to *linestring* in the physical schema, but have completely different behaviors in the conceptual model.

4 AST-POSTGIS

In this article, 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/pgSQL[12], AST-PostGIS is currently available for PostgreSQL version 9.5 or above and requires the spatial extension 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 special table. In order to discern these extension objects from those already implemented in PostgreSQL and PostGIS, AST-PostGIS adopts as a standard the **AST** prefix for all names.

AST-PostGIS is intended to reduce the distance 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 additional control, respecting the semantics for geo-object 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 establishing 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

[12]PL/pgSQL is a loadable procedural language for the PostgreSQL DBMS.

identify inconsistent data, and indicate constraint violations in a log table. The following subsections explain, in detail, each of the features introduced by AST-PostGIS.

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 the column type in 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 functions of PostGIS.

Table 2 lists the 11 Advanced Spatial Data Types implemented in AST-PostGIS, along with the corresponding OMT-G classes from which they derive, the PostGIS primitive types that materialize their basic geometry, and the integrity constraints that control their behavior, defined formally in Section 3.2. These integrity constraints are implemented in PL/pgSQL scripts and encapsulated in the extension.

4.2 Integrity constraints for spatial relationships

AST-PostGIS provides integrity constraints for 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 spatial relationship 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, if 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 triggers to execute one of the custom procedures, to be configured to fire **after** an event occurs, and to run **for each statement**. The table associated with the trigger must be one of the tables involved in the relationship. In case of a spatial aggregation, the associated table must be the one that represents the *part* of the whole-part relationship. For arc-node relationships, the associated table can be either the *arc* or the *node*. If it is the table containing arcs, the trigger blocks arc insertions or updates if there are not two nodes connected to them. If the associated table contains nodes, the trigger blocks nodes that are 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 the way 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 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 parameters. 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. Listing 1 shows how a trigger for a spatial aggregation can be created.

```
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 1: Aggregation Trigger

The procedure *ast_arcnodenetwork* is polymorphic and admits two configurations for the parameters. When used with an arc-node network, the procedure requires four 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 the table that represents the nodes, and (4) the name of the geometry column of the node table, whose type is *ast_node*. The configuration of a trigger and its procedure to manage an arc-node network is illustrated in Listing 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 geometry column.

```
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 2: Arc-Node Trigger

AST-PostGIS supports the minimum set of spatial relationship operators, identified by Clementini et al. [11] and adopted by OMT-G [13], from which all others can be specified: **Crosses**, **Disjoint**, **Overlaps**, **Touches** and **Within**. Besides them, the extension considers a larger set of spatial relationships for convenience, based on the DE-9IM standard. This includes relationships such as **Contains**, **ContainsProperly**, **Covers**, **CoveredBy** and **Intersects** In addition, two metric 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 3). This trigger must 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

Table 2: 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_planarsubdivision	planar subdivision	geometry(polygon)	R1, R4
ast_tin	triangular irregular network	geometry(polygon)	R1, R5
ast_tesselation	tesselation	raster	R1, R3
ast_sample	sample	geometry(point)	-
ast_uniline	unidirectional line	geometry(linestring)	R6
ast_biline	bidirectional line	geometry(linestring)	R6

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, violating rule R9. This problem is addressed 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, and configured to fire only after **delete** events for arc-node networks and spatial relationships, and **update** and **delete** events for spatial aggregations.

```
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 3: Spatial Relationship Trigger

4.3 Consistency Check Functions

The spatial integrity constraints introduced in Section 4.2 have to be applied when a database is created and before any data is stored. They work by checking data operation events (insertions, updates and deletions) as they occur. AST-PostGIS provides consistency check functions, as shown in Table 3, to verify a non-empty spatial database for existing violations on spatial relationships. 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. Consistency check functions also require parameters, which are the same as in the procedures described in last subsection.

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

Listing 4: Example of consistency check function call

The *ast_violation_log* table is used by the consistency check functions to record information about the inconsistencies encountered on the spatial relationships. The log table was designed to store the necessary information for the solution of inconsistencies, including the type of relationship that was violated, the rows related to the error, and the geometry that causes the error.

4.4 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.

The trigger procedures, introduced by our extension to add integrity constrains 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.

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 section. The schema was composed in an online interactive design tool, OMT-G Designer [24] that is capable of automatically mapping the OMT-G diagram

Table 3: Consistency check functions supported by AST-PostGIS

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

to PostGIS using AST-PostGIS, generating a complete set of DDL commands and triggers in a script.

5.1 Conceptual schema

Figure 5 shows part of the conceptual schema for an urban cadastral database system. The class diagram includes most of the primitives defined in OMT-G. The diagram corresponds to the geographic area of a municipality. The city can be represented as a point or as a polygon (its boundaries). The boundaries contain city blocks, which are in turn subdivided into parcels. Each parcel is represented by its polygonal boundary and can be occupied by residential, commercial or industrial buildings. Due to environmental regulations, industries cannot operate within 800 meters of nature reserves. Street addresses are formed by concatenating the thoroughfare name to the street number. Each address is represented by a point, and is to be located inside the parcel. A thoroughfare is modeled as a set of street segments, which compose the arcs in a street network. Thoroughfare intersections are represented by nodes at the crossings. The municipality's space is also entirely subdivided into neighborhoods. In addition, relief is represented by a set of isolines which, as a geo-field, cover the entire municipal territory.

Several spatial integrity constraints can be derived from this schema. Table 4 lists the constraints that must be observed in the physical implementation.

5.2 Physical implementation

The SQL snippet in Listing 5 shows how part of the conceptual schema illustrated in Figure 5 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

Table 4: Spatial integrity constraints derived from the schema in Figure 5

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`
RX	`Block` *within* `City boundary`, `Building` *within* `Parcel`, `Parcel` *contains* `Address`, `Nature reserve` *distant(800)* `Industry`

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 *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.

```sql
-- 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 (
  paviment varchar(10),
  thoroughfare varchar(50) REFERENCES Thoroughfare(name),
  geom ast_biline
);
```

Listing 5: SQL schema for the tables definition

The code snippet in Listing 6 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 names 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

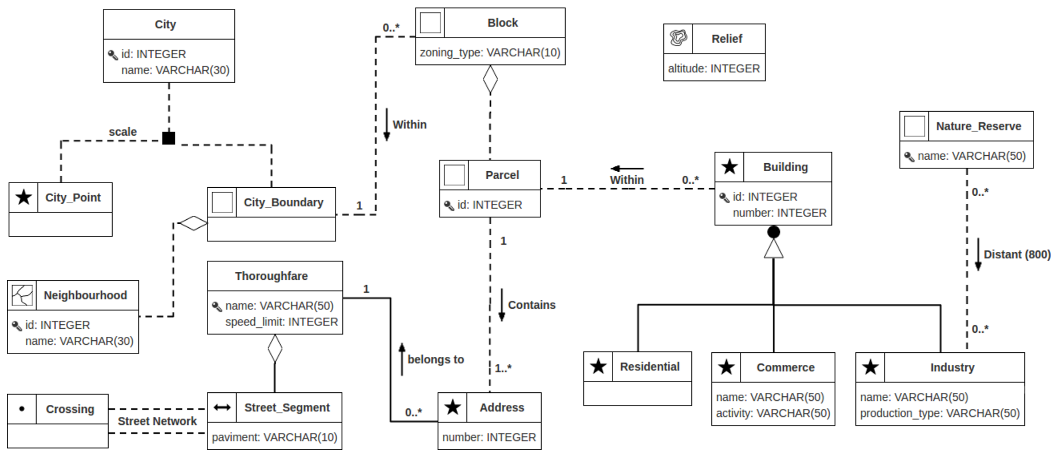


Figure 5: OMT-G class diagram for an urban geographic application

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* also requires the value of the distance as a parameter.

```
-- 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 6: Integrity constraints for aggregations

Listing 7 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, and that a parcel cannot be created without an address. Exceptions are raised by these triggers if a block is not created within city boundaries, or if an industry object is positioned without observing the clearance distance of nature reserves, blocking invalid updates.

```
-- 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 7: Integrity constraints for spatial relationships

Lastly, the trigger in Listing 8 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 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.

```
-- 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 8: Integrity constraint for arc-node network

6 CONCLUSIONS

This paper 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 deletions. 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 RDBMSs, 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 RDMBS 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.

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. Another possible outcome is the design of specific data structures in the DBMS to support the implementation of some of the more computationally demanding integrity checks, such as planar subdivision, without resorting to triggers. Thus, the backend of AST-PostGIS could be reimplemented using more efficient methods, in support of the proposed advanced spatial data types.

ACKNOWLEDGMENTS

The authors acknowledge the support of CNPq and FAPEMIG, Brazilian agencies in charge of fostering research and development.

REFERENCES

[1] Serge Abiteboul and Richard Hull. 1987. IFO: a formal semantic database model. *TODS: ACM Transactions on Database Systems* 12, 4 (1987), 525–565.

[2] Graça Abrantes and Rogério Carapuça. 1994. Explicit representation of data that depend on topological relationships and control over data consistency. In *Proceedings of the 5th European Conference and Exhibition on Geographical Information Systems, EGIS/MARI '94*. Utrecht : EGIS Foundation, Paris, France, 869–877.

[3] David W. Adler. 2001. DB2 Spatial Extender – spatial data within the RDBMS. In *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*. Morgan Kaufmann Publishers Inc., Roma, Italy, 687–690.

[4] Alastair Aitchison. 2009. *Beginning Spatial with SQL Server 2008*. Apress.

[5] Yvan Bédard, Claude Caron, Zakaria Maamar, Bernard Moulin, and Denis Vallière. 1996. Adapting data models for the design of spatio-temporal databases. *Computers, Environment and Urban Systems* 20, 1 (1996), 19–41.

[6] Grady Booch, James Rumbaugh, and Ivar Jacobson. 2005. *The Unified Modeling Language User Guide* (2nd ed.). Addison-Wesley Professional.

[7] Karla A. V. Borges, Clodoveu A. Davis Jr., and Alberto H. F. Laender. 2001. OMT-G: an object-oriented data model for geographic applications. *GeoInformatica* 5, 3 (2001), 221–260.

[8] Karla A. V. Borges, Clodoveu A. Davis Jr., and Alberto H. F. Laender. 2002. Integrity Constraints in Spatial Databases. In *Database Integrity: Challenges and Solutions*, Jorge Horacio Doorn and Laura C. Rivero (Eds.). Idea Group, Chapter 5, 144–171.

[9] Karla A. V. Borges, Clodoveu A. Davis Jr., and Alberto H. F. Laender. 2005. Modelagem conceitual de dados geográficos. In *Banco de dados geográficos*, Marco Antonio Casanova, Gilberto Câmara, Clodoveu A. Davis Jr., Lúbia Vinhas, and Gilberto Ribeiro Queiroz (Eds.). MundoGEO, Chapter 3, 93–146.

[10] Peter Pin-Shan Chen. 1976. The entity-relationship model – toward a unified view of data. *TODS: ACM Transactions on Database Systems* 1, 1 (1976), 9–36.

[11] Eliseo Clementini, Paolino Di Felice, and Peter Van Oosterom. 1993. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In *Advances in Spatial Databases: 3rd International Symposium, SSD '93 (Lecture Notes in Computer Science)*, Vol. 692. Springer, Singapore, 277–295.

[12] Clodoveu A. Davis Jr., Karla A. V. Borges, and Alberto H. F. Laender. 2001. Restrições de integridade em bancos de dados geográficos. In *Proceedings of the 3rd Brazilian Workshop on GeoInformatics, GEOINFO 2001*. Instituto Nacional de Pesquisas Espaciais (INPE), Rio de Janeiro, Brazil, 63–70.

[13] Clodoveu A. Davis Jr., Karla A. V. Borges, and Alberto H. F. Laender. 2005. Deriving Spatial Integrity Constraints from Geographic Application Schemas. In *Encyclopedia of Database Technologies and Applications*, Laura C. Rivero, Jorge Horacio Doorn, and Viviana E. Ferraggine (Eds.). Idea Group, 176–183.

[14] Yi Fang, Marc Friedman, Giri Nair, Michael Rys, and Ana-Elisa Schmid. 2008. Spatial indexing in Microsoft SQL Server 2008. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data (SIGMOD '08)*. ACM, New York, USA, 1207–1216.

[15] Vincenzo Del Fatto, Vincenzo Deufemia, Luca Paolino, and Sara Tumiati. 2015. WiSPY: A Tool for Visual Specification and Verification of Spatial Integrity Constraints. *VLSS: Journal of Visual Languages and Sentient Systems* 1 (2015), 39–48.

[16] Andrew U. Frank and David M. Mark. 1991. Language issues for geographical information systems. In *Geographical Information Systems: Principles and Applications*, D. J. Maguire, M. F. Goodchild, and D. W. Rhind (Eds.). Vol. 1. Longman Scientific and Technical, Chapter 11, 147–163.

[17] André C. Hora, Clodoveu A. Davis Jr., and Mirella M. Moro. 2010. Generating XML/GML schemas from geographic conceptual schemas. In *AMW 2010, Proceedings of the 4th Alberto Mendelzon International Workshop on Foundations of Data Management (CEUR Workshop Proceedings)*, Vol. 619. CEUR-WS.org, Buenos Aires, Argentina.

[18] 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. https://www.iso.org/standard/60343.html

[19] Karen K. Kemp. 1992. *Environmental modeling with GIS: a strategy for dealing with spatial continuity.* Ph.D. Dissertation. University of California, USA.

[20] Georg Kosters, Bernd-Uwe Pagel, and Hans-Werner Six. 1997. GIS-application development with GeoOOA. *IJGIS* 11, 4 (1997), 307–335.

[21] Jugurta Lisboa Filho and Cirano Iochpe. 1999. Specifying Analysis Patterns for Geographic Databases on the Basis of a Conceptual Framework. In *Proceedings of the 7th ACM International Symposium on Advances in Geographic Information Systems, GIS '99*. ACM, Kansas City, USA, 7–13.

[22] Jugurta Lisboa Filho, Mauricio Fidelis Rodrigues Jr., Jaudete Daltio, and Victor de Freitas Sodré. 2004. 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*. Ostrava, Czech Republic, 29–36.

[23] Jugurta Lisboa Filho, Victor Freitas Sodré, Jaudete Daltio, Maurício Fidelis Rodrigues Jr., and Valério Moysés Vilela. 2004. A CASE Tool for Geographic Database Design Supporting Analysis Patterns. In *Proceedings of Conceptual Modeling for Advanced Application Domains, ER 2004 Workshops CoMoGIS, COMWIM, ECDM, CoMoA, DGOV, and ECOMO (Lecture Notes in Computer Science)*, Vol. 3289. Springer, Shanghai, China, 43–54.

[24] Luís Eduardo Oliveira Lizardo and Clodoveu A. Davis Jr. 2014. OMT-G Designer: a Web tool for modeling geographic databases in OMT-G. In *Advances in Conceptual Modeling: 33rd International Conference on Conceptual Modeling, ER 2014 (Lecture Notes in Computer Science)*, Vol. 8823. Springer, Atlanta, USA, 228–233.

[25] Jim Melton and Andrew Eisenberg. 2001. SQL multimedia and application packages (SQL/MM). *ACM SIGMOD Record* 30, 4 (2001), 97–102.

[26] Bruce Momjian. 2001. *PostgreSQL: Introduction and Concepts*. Addison-Wesley.

[27] Regina O. Obe and Leo S. Hsu. 2015. *PostGIS in action* (2nd ed.). Manning.

[28] 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.

[29] 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.

[30] Juliano Lopes Oliveira, Fátima Pires, and Claudia Bauzer Medeiros. 1997. An environment for modeling and design of geographic applications. *GeoInformatica* 1, 1 (1997), 29–58.

[31] Oracle 2017. *Spatial and Graph Analytics with Oracle Database 12c Release 2.* Technical White Paper. Oracle Corporation.

[32] Adam Piórkowski. 2011. MySQL Spatial and PostGIS – implementations of spatial data standards. *EJPAU* 14, 1 (2011), 1–8.

[33] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William E. Lorensen, et al. 1991. *Object-Oriented Modeling and Design.* Prentice Hall.

[34] Shashi Shekhar, Mark Coyle, Brajesh Goyal, Duen-Ren Liu, and Shyamsundar Sarkar. 1997. Data models in geographic information systems. *Commun. ACM* 40, 4 (1997), 103–111.

[35] Knut Stolze. 2003. SQL/MM Spatial – The Standard to Manage Spatial Data in a Relational Database System. In *Tagungsband der 10. BTW-Konferenz, BTW 2003 (LNI)*, Vol. 26. GI, Leipzig, Germany, 247–264.

[36] Michael Widenius and Davis Axmark. 2002. *MySQL Reference Manual: Documentation from the Source.* O'Reilly Media.

[37] Michael F. Worboys, Hilary M. Hearnshaw, and David J. Maguire. 1990. Object-oriented data modelling for spatial databases. *IJGIS* 4, 4 (1990), 369–383.