Building a Robust Relational Implementation of Topology

Erik Hoel, Sudhakar Menon, and Scott Morehouse

Environmental Systems Research Institute
380 New York Street
Redlands, CA 92373
{ehoel,smenon,smorehouse}@esri.com

Abstract. Topologically structured data models often form the core of many users' spatial databases. Topological structuring is primarily used to ensure data integrity; it describes how spatial objects share geometry. Supporting topology within the context of a relational database imposes additional requirements – the complex topological model must retain integrity across transactional boundaries. This can be a problematic requirement given the complexities associated with implementing safe referential integrity structures in relational databases (e.g., bulk data loading into a topologically structured model) [19, 5]. Common implementation techniques such as allowing dangling pointers (i.e., null foreign keys) complicates the issues for client applications that consume these models. In this paper, we revisit the problem of building a robust and scalable relational implementation of a topologically structured data model. We propose a different approach to representing such models that avoids many of the traditional relational database problems associated with maintaining complex semantic models.

1 Introduction

Topological data structures have been used to represent geographic information for over thirty years [7, 24]. The topological model has been the basis of a number of operational systems (see, for example TIGER/db [4], ARC/INFO [22], or TIGRIS [14]). These systems have been based on binary file and in-memory data structures and support a single-writer editing model on geographic libraries organized as a set of individual map sheets or tiles.

Recent developments in geographic information systems have moved to database-centered information models. One aspect of this work has been to replace the file system with a relational database engine as the persistence mechanism for geographic information. However, replacement of the physical I/O layer is only one aspect to consider when designing a "database GIS". Other aspects of the database concept must also be considered.

- What is the generic spatial information model and what are the associated spatial operators?
- How does the design support multiple, simultaneous writers?
- How is the semantic integrity of the data model declared, maintained, and enforced?
- How to design a system which performs well and can scale to support hundreds of simultaneous users?

Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.

- How to design a system which can support very large continuous spatial databases, containing millions of geographically interrelated objects (e.g., the road network of the United States or the land ownership fabric of Austria)?

Database GIS involves more than simply exporting classic GIS data structures to a normalized relational schema. Database GIS must also address the functional aspects of being a database as well: the integrity model, the transaction model, and performance/scalability issues. Otherwise database GIS implementations run the risk of becoming "off-line data repositories" (at best) or "write-only databases" (at worst).

The representation of topological data models using relational database concepts and engines should be of interest to the database research community as well as to geographic information scientists. The data model serves as an excellent case study for considering problems of information modeling and, especially, for evaluating architectures for the management of semantic integrity.

In this paper, we describe a design for modeling GIS topology using relational concepts and database engines. This design is the basis for our implementation of topology in the ArcGIS geographic information system.

In the first section, we introduce the logical model of GIS topology. We then consider a physical database implementation using the conventional notions for mapping entities and relationships to tables and the conventional primary key / foreign key referential integrity model. Problems in this approach are discussed. We then present an alternative implementation of the topology model which uses an unconventional approach to the problem of database integrity and transaction management.

2 What is GIS Topology?

Topological data structures for representing geographic information are a standard topic in geographic information science (see [Güti95], for example, for an excellent definition of the mathematical theory underlying this information model). In general, the topological data model represents spatial objects (point, line, and area features) using an underlying set of topological primitives. These primitives, together with their relationships to one another and to the features, are defined by embedding the feature geometries in a single planar graph. Such datasets are said to be "topologically integrated".

The model associates one or more topological primitives (i.e., nodes, edges, and faces [16]; or 0-cells, 1-cells, and 2-cell in the TIGER parlance [3, 8]) with spatial objects of varying geometry type (i.e., points, lines, and polygons respectively). More specifically, a feature with point geometry is associated with a single node element, a feature with line geometry is associated with one or more edge elements, and a feature with polygon geometry is associated with one or more face elements. This is depicted in Fig. 1 as the generic topology model.

Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.

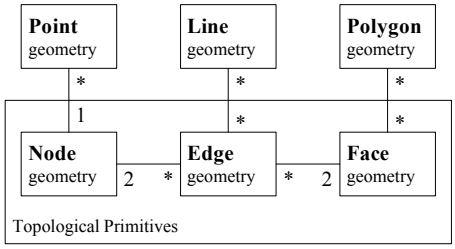
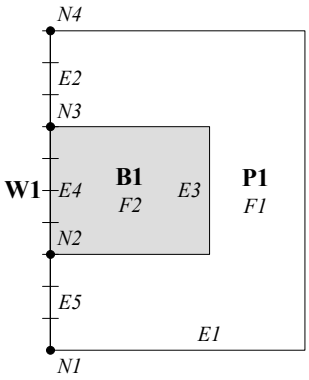


Fig. 1. Generic topology model

There are additional relationships between the topological elements themselves as is also shown in Fig. 1. A node element may or may not be associated with a collection of edge elements. A face element may be associated with one or more edge elements. Finally, an edge element is associated with two node elements and two face elements. The relationships between nodes and faces may either be implicit or explicit. We have represented these relationships between nodes and faces as implicit within Fig. 1.

A concrete example, showing a specific instance of this model is shown in Fig. 2. The database consists of three classes which represent real geographic entities: Parcels, Walls, and Buildings. In the example, there is one instance of each class. In practice, these classes could contain millions of instances. The wall and the building are coincident with the western boundary of the parcel as shown.



Parcels

ID	*Vertices*	*Faces*	*Edges*
P1	(0,0),(0,3),(0,7),(0,8),(8,8),(8,0),(0,0)	F1, F2	-E5,+E4,-E2,+E1

Buildings

ID	*Vertices*	*Faces*	*Edges*
B1	(0,3),(0,7),(5,7),(5,3),(0,3)	F2	+E3,+E4

Walls

ID	*Vertices*	*Edges*
W1	(0,10),(0,7),(0,3),(0,0)	+E2,-E4,+E5

Nodes

ID	*Vertex*	*Edges*	*Features*
N1	(0,0)	-E1,-E5	-
N2	(0,3)	+E4,-E3,+E5	-
N3	(0,7)	-E2,-E3,-E4	-
N4	(0,10)	+E1,+E2	-

Edges

ID	*Vertices*	*From*	*To*	*Left*	*Right*	*Features*
E1	(0,10),(8,10),(8,0),(0,0)	N4	N1	F0	F1	-
E2	(0,10),(0,7)	N4	N3	F1	F0	W1
E3	(0,7),(5,7),(5,3),(0,3)	N3	N2	F1	F2	-
E4	(0,3),(0,7)	N2	N3	F1	F2	W1
E5	(0,3),(0,0)	N2	N1	F1	F0	W1

Faces

ID	*Vertices*	*Edges*	*Features*
F0	(0,0),(8,0),(8,10),(0,10),(0,0)	+E2,-E4,+E5,-E1	-
F1	(0,10),(8,10),(8,0),(0,0),(0,3),(5,3),(0,7),(0,10)	+E1,-E5,-E3,-E2	P1
F2	(0,7),(5,7),(5,3),(0,3),(0,7)	+E3,+E4	P1,B1

Fig. 2. An instance of the generic topology model

Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.

It is interesting to note that the objects have both set and list-based properties. For example, the geometry of an edge is defined by a list of coordinates and the object references to the edges participating in a line feature (e.g., W1) are ordered and oriented to properly represent the orientation of the linear feature. Edges and vertices of polygons are oriented clockwise (with the interior of the polygon on the right). For clarity, we have showed attributes, such as object geometries, redundantly in multiple object classes. In some physical implementations of the model, these geometries would only be stored once, on edge primitives for example, and instantiated via queries for other features.

After examining this example, it is clear that the logical topology model is a complex graph model, containing ordered object associations based on the geometric embedding of the objects in two-dimensional space. It should also be clear that common queries, such as "draw a map of all buildings" or "find features contained within a given parcel" require ordered navigations of relationships, not simply set operations.

Given the inherent complexity of this representation, it is important to reflect upon why people want topology in their spatial datasets in the first place – i.e., what are the fundamental requirements. At a high level, topology is employed in order to:

- Manage shared geometry (i.e., constrain how features share geometry).
- Define and enforce data integrity rules (e.g., no gaps between features, no overlapping features, and so on).
- Support topological relationship queries and navigation (e.g., feature adjacency or connectivity).
- Support sophisticated editing tools (tools that enforce the topological constraints of the data model).
- Construct features from unstructured geometry (i.e., polygons from lines).

The logical topology model provides a theoretical basis for this functionality. For example, the constraint "buildings must not overlap one another" can be expressed by the topology constraint "faces may only be associated with a single feature of type building". Similarly, the problem of creating polygons from unstructured lines can be stated as: "calculate edges, faces, and nodes from lines; create a feature on top of each resulting face".

In GIS, topology has historically been viewed as a physical spatial data structure that directly implements the objects of the logical topology model. However, it is important to realize that this physical data structure is only useful because it is a tool for data integrity management, spatial queries / navigation and other operators. It is possible to consider alternate implementations of the logical topology model which also support this functionality. In effect, topology must be considered as a complete data model (objects, integrity rules, and operators), not simply as storage format or set of record types.

3 Conventional Physical Implementation

The logical topology model can be implemented for relational database engines in a straight forward fashion as a normalized relational model with explicit representation of topological primitives using keys (primary and foreign) to model the topological relationships (see Fig. 3). We can call this the conventional relational topology model.

Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.

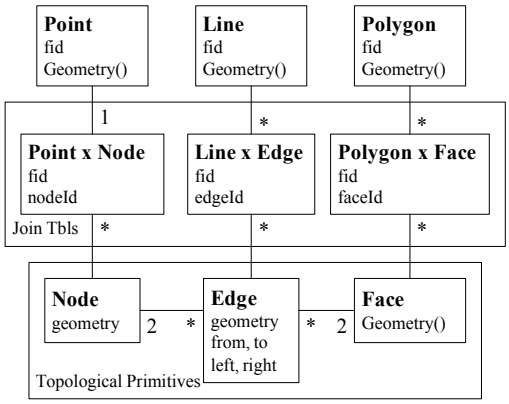


Fig. 3. Conventional relational topology model with join tables (e.g., Line x Edge)

This implementation uses referential integrity constraints [13, 20] to declare integrity rules for the topology. Join tables are employed to support many-to-many relationships between the features and their associated topological primitives (in databases which support array types, the join tables may be replaced by lists of foreign keys embedded in the feature and primitive tables). In addition, the geometry of the features is normalized - that is, it is assembled from their associated topological primitives. Fig. 4 illustrates our sample dataset example in this model.

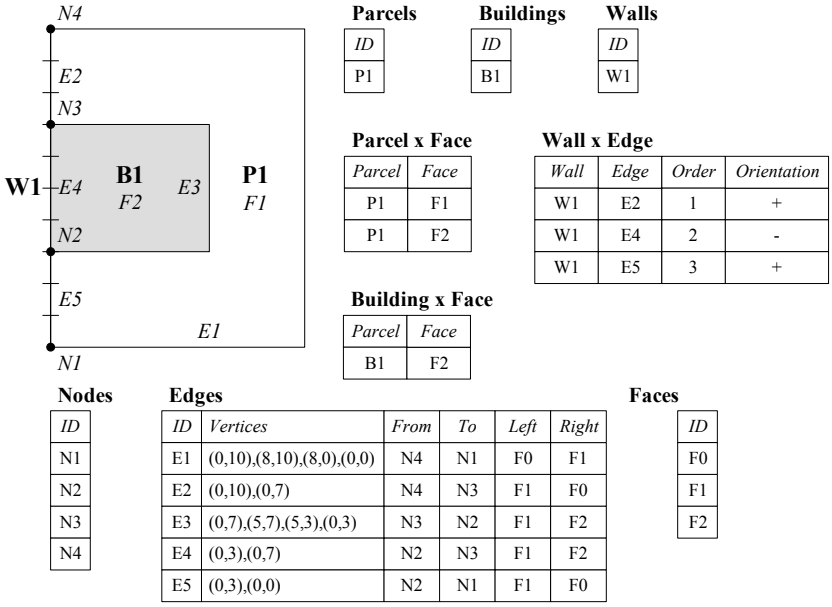


Fig. 4. An example instance of the conventional relational implementation

There are a number of advantages to this physical implementation. First, there is a direct and rather trivial mapping to the logical topology model. There is conceptual comfort for many users because all of the topological primitives are explicitly persisted, as are many of

Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.

the primitive topological relationships. The geometry itself is only represented once within the database. Finally, there is a degree of client independence given that the topological primitives and relationships can be edited using standard SQL updates.

However, there are a number of serious problems with this implementation, these relate to:

- the performance of typical queries,
- the maintenance of semantic integrity in the model, and
- the performance and complexity of typical updates.

3.1 Query Performance

Although this implementation eliminates redundant storage of geometric information for features, complex queries are required to instantiating the features' geometry (the most common query for most GIS software) [28]. For example, to fetch the geometry of the parcel P1, a query must combine information in four tables (Parcels, Parcel x Face, Faces, and Edges) and use external geometric/topological logic to correctly assemble the geometry of the Parcel. When this query must be performed iteratively – to draw all parcels, for example – we end up nesting complex queries or executing nested cursors in the application code. In general, navigational access to relationships (dereferencing pointers) is significantly slower with relational technology that with other technologies (e.g., object databases [17]).

3.2 The Integrity Mechanism

A more fundamental problem relates to the integrity of the database. Despite the use of (and paying the performance cost for) referential integrity constraints, the integrity and consistency of the topology data model cannot actually be defined using the conventional referential integrity model! Referential integrity constraints simply declare that "if a foreign key is not null, then the referenced primary key exists". This is a very weak (one might almost say meaningless) basis for managing the integrity of the topological data model. For a consistent state of the data model, there must be no null references and the geometry of the primitives must be consistent with the objects and references. The conventional referential integrity model has no means for declaring such "semantic" constraints, much less enforcing them.

The integrity of a topology is a global property of the set of entities comprising the topology, rather than constraints or behavior attached to individual entities. Topology integrity must be validated by considering a complete set of updates as a whole. This validation logic must analyze the geometric configuration of features in the changed area to ensure that the topology primitives and relationships are correct. This validation logic executes externally to the core relational integrity model.

It is possible to execute arbitrary business logic in response to data updates using triggers. Such a mechanism could be used to accumulate change information, and then execute global validation logic. However, active relational databases (i.e., those supporting constraint and trigger mechanisms) continue to have lingering implementation and performance drawbacks [5, 26]. Database extensions that are implemented as trigger-based services (that may or may not be automatically generated) where the complexity of the

Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.

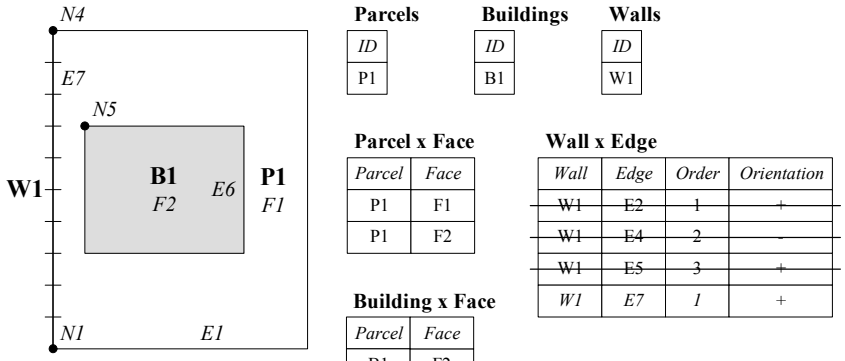
domain is non-trivial (e.g., business rules, supply chain management, rule-based inference systems, and topology) suffer from several fundamental problems:

- difficulty of implementation (subtlety of behavior, primitive debugging tools),
- performance problems and scalability with complex trigger collections (lack of sophistication of trigger processors), and
- lack of uniformity (portability between RDBMSs, though this will be helped by the SQL-99 standard [11]).

For these reasons, complex semantic behavior (also known as the business objects) is more commonly implemented in shared code running in an application server tier.

3.3 Complexity of Updates

Simple edits to a topological data model can result in a complex sequence of changes to the associated topology primitives and relationships. Consider a simple edit to the sample dataset of Fig. 4: let's move the building one unit to the right. The changes necessary in the data model are shown in Fig. 5.



Parcels

ID
P1

Buildings

ID
B1

Walls

ID
W1

Parcel x Face

Parcel	*Face*
P1	F1
P1	F2

Wall x Edge

Wall	*Edge*	*Order*	*Orientation*
~~W1~~	~~E2~~	~~1~~	~~+~~
~~W1~~	~~E4~~	~~2~~	~~-~~
~~W1~~	~~E5~~	~~3~~	~~+~~
W1	*E7*	*1*	+

Building x Face

Parcel	*Face*
B1	F2

Nodes

ID
N1
~~N2~~
~~N3~~
N4
N5

Edges

ID	*Vertices*	*From*	*To*	*Left*	*Right*
E1	(0,10),(8,10),(8,0),(0,0)	N4	N1	F0	F1
~~E2~~	~~(0,10),(0,7)~~	~~N4~~	~~N3~~	~~F1~~	~~F0~~
~~E3~~	~~(0,7),(5,7),(5,3),(0,3)~~	~~N3~~	~~N2~~	~~F1~~	~~F2~~
~~E4~~	~~(0,3),(0,7)~~	~~N2~~	~~N3~~	~~F1~~	~~F2~~
~~E5~~	~~(0,3),(0,0)~~	~~N2~~	~~N1~~	~~F1~~	~~F0~~
E6	*(1,7),(6,7),(6,3),(1,3),(1,7)*	*N5*	*N5*	*F1*	*F2*
E7	*(0,10),(0,0)*	*N4*	*N1*	*F1*	*F2*

Faces

ID
F0
F1
F2

Fig. 5. An example instance of the conventional relational implementation. In this example, deleted rows are indicated with the horizontal strike through; new rows are in italics

This edit to the geometry of a single feature requires that we validate the geometry and topology of the effected area. The edit requires updates and inserts to multiple tables: Edges, Nodes, and Edges x Walls. These updates must either be made correctly by the application prior to executing the validation logic.

Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.

The example also raises an interesting point related to the "lifecycle semantics" of topological primitives. These spatial objects do not correspond to any feature in either the real or legal-boundary worlds [25]. They exist solely as constructs of the topology for a given configuration of features. How is "identity" defined for such objects? In the example, did we replace edges E2, E4, and E5 with E7 or should we have updated one of the existing edges? These lead to additional complexity when we deal with identifier generation, historical data management, and other factors. By making topological primitives explicit entities in the model, we have another set of business objects to manage.

Update operations on the explicit topology implementation are complex (irrespective of whether stored procedures/triggers or client-side code is employed). This complexity is closely related to performance concerns when persisting topological primitives in relational databases. The performance difference can be quite significant when compared with existing file-based topology solutions (e.g., orders of magnitude). It is important to note that the increased server side processing required with the complex trigger mechanisms will impact server scalability [5].

These considerations, reinforced by practical experience implementing the normalized model using prototype software led to the development of an alternative approach, described in the following section.

4 Alternative Physical Implementation

In order to address some of the problems inherent in the conventional topology physical implementation, we describe a new model that is currently hosted within the ArcGIS Geodatabase [10, 32]. With this model, we make three fundamental departures. First, we relax the conventional transactional model and allow the incremental validation of the topology (i.e., validation is performed as a bulk process at certain user-defined intervals or events). Thus, features with geometry that has not been topologically validated can be maintained within the model. This has a significant impact upon the user's editing experience where each individual edit operation does not need to topologically restructure the area being edited. Second, we cache geometry in the features rather than only in the topological primitives in order to eliminate relationship navigation for common queries (e.g., drawing the features). Third and finally, we use logic external to the relational database for geometric and topological validation.

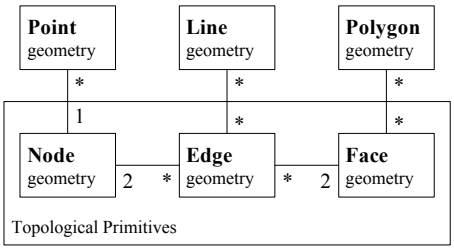


Fig. 6. Geodatabase topology model

It is possible in the generic model (r.e., Fig. 1) to obtain topological primitives from feature geometry; similarly, it is possible to obtain feature geometry from topological

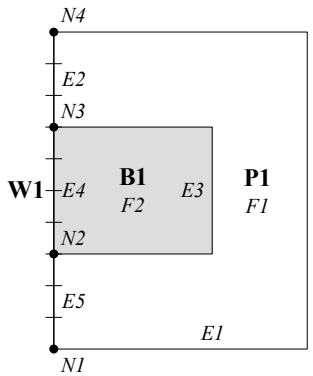
Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.

primitives [28]. Effectively, the geometry found in features is a dual representation of the geometry that would be found on the topological primitives. We have chosen to simplify and streamline the generic explicit topology model and to not persist both representations.

The topological primitives are not persisted as a special type of feature; instead, feature geometry is persisted and topological primitives (along with their geometry) are inferred. The process of topological integration (validation) results in vertex equality where features share underlying topological primitives. Given vertex equality, reconstruction of topological primitives is straight forward. Vertices on feature geometries in this scheme play the same role as that assigned to embedded foreign keys in data structures that explicitly model topological primitives.

Topological primitives and relationships are only instantiated during the process of topological validation or when required by the client application (note that this aspect is similar to MGE where topology is selectively built but the topological primitives are not persisted in the RDBMS [15]). The primary reason for this alternative approach is that it is easier (faster, more scalable) to recreate an index (e.g., the topological primitives) than to do all the bookkeeping necessary to persist and retrieve it from the database while preserving the database transaction model (note that we have also found the same to be true when modeling networks as well as surfaces – i.e., TINs). Additionally, it is frequently the case that the portion of the topological primitives necessary for an operation is small relative to the entire topology (e.g., editing a few block groups in a localized area within TIGER).

It is important to note that for this approach to be viable from a performance standpoint, it is critical that there exist a high performance topology engine that validates the portion of the topology in question (see the description below) as well as instantiate the topological primitives for the given collection of features within the topology [29, 30].



Parcels

ID	*Vertices*
P1	(0,0),(0,3),(0,7),(0,10),(8,10),(8,0),(0,0)

Buildings

ID	*Vertices*
B1	(0,3),(0,7),(5,7),(5,3),(0,3)

Walls

ID	*Vertices*
W1	(0,10),(0,7),(0,3),(0,0)

Topology Rules

ID	*Rule type*
R1	Parcels no overlap
R2	Buildings no overlap

Topology Errors

ID	*Vertices*	*Rule type*	*Feature*	*IsException*
E1	...	-	-	-

Dirty Area

Vertices
...

Fig. 7. An example instance of the Geodatabase topology implementation

At a high level, topology within the Geodatabase consists of a collection of feature classes (homogeneous collections of features), topology rules, and other metadata used to support the validation model. This metadata includes dirty areas, topology errors, and the cluster tolerance. An example instance of the topology implementation is shown in Fig. 7.

4.1 Topology Rules

Topological integrity is defined with respect to a collection of topology rules. *Topology rules* are used to define constraints on the permissible topological relationships between features in one or more feature classes that participate in the topology. Topology rules are considered part of the topology metadata; they are not considered metadata associated with the feature classes that participate in the topology. The collection of topology rules that are associated with the topology are selected on the basis of which topological relationships are important for the user's model. There is no fixed set of topology rules that are associated with all topologies; instead, topologies may be specified with zero or more rules (note that one significant utility of a topology without any topology rules is that Milenkovic's third normalization rule – see Section 3.2 - are enforced on the features as a byproduct of validation).

Topology rules are checked when the topology is validated. When a topology rule is violated, a topology error object is generated. This topology error may be represented as a special type of feature that may itself be persisted. At a later point following the validation, the user may then review the topology error objects and the error conditions may be corrected. Topology rule violations do not prevent the validation operation from successfully completing.

Examples of topological rules that may be applied to polygon features include:

- The interiors of polygons in a feature class must not overlap (they may however share edges or vertices).
- Polygons must not have voids within themselves or between adjacent polygons (they may share edges, vertices, or interior areas).
- Polygons of one feature class must share all their area with polygons in another feature class (i.e., they must cover each other).
- The boundaries of polygon features must be covered by lines in another feature class.

There are of course numerous other topology rules that may be specified for each of the different geometry types. Note that it would also be possible for a system to be designed where all topology rules were specified using Clementini relationships [6] or the 9-Intersection model [9].

4.2 Validation

The validation process is a fundamental operation on a topology performed by a topology engine [29, 30]. The validation process on a topology is responsible for ensuring Milenkovic's third normalization rule [21] on all spatial objects participating in the topology are respected (i.e., no intersecting edges, no edge endpoints within the tolerance, no edge endpoints within the tolerance of another edge). In addition, the validation process is responsible for checking all specified topology rules and generating topology errors at locations where rules are violated.

The basic processing flow for the validation process within the topology engine is as follows:

- Load all the feature geometries and associated topology metadata (topology rules, feature class weights, and cluster tolerance).

Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.

- Crack, cluster, classify, and topologically structure the nodes and edges.
- Create new topology error instances when topology rules are found to be in violation. Delete pre-existing error instances if the rules are no longer in violation.
- Update the feature geometries as necessary (i.e., if their geometries were modified during the establishment of the Milenkovic conditions).
- Update the dirty areas associated with the topology.

It is important to note that the validation process does not need to span all features within the topology dataset. A validation can be performed on a subset of the space spanned by the dataset. This is a complex task given that it will require the validation of topology rules using partial information (e.g., certain error instances may only be partially contained within the region being validated).

4.3 Dirty Areas

A topology can have an associated *dirty area* – a dirty area corresponds to the regions within the topology extent where features participating in the topology have been modified (added, deleted, or updated) but have yet to be validated. When the geometry of a feature that participates in a topology is modified, the extent of the dirty area is enlarged to encompass the extent of the bounding rectangle of the modified geometry (note that other simplified geometry representations may also be employed - e.g., convex hulls). This is depicted in Fig. 8. The dirty area is persisted with the topology. In order to ensure that the topology is correct, the topology in the dirty areas will need to be validated.

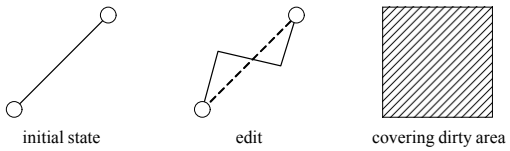


Fig. 8. Example of dirty area creation following a feature edit. The dirty area is depicted by the hatched square

It is not necessary to validate the entire space spanned by the dirty area at one time; instead, a subset of the dirty area can be validated. If the dirty area is partially validated, the original dirty area will be clipped by the extent of the region that is validated.

Allowing users the ability to validate a portion of the dirty area is a pragmatic requirement of supporting extremely large seamless topologies – for example, when a topology is first created, or when the topology metadata (e.g., topology rules, etc.) is modified, the entire extent of the topology is dirty. If users were not provided with the capability to validate a portion of the dirty area, the user would be required to validate the entire topology which could prove to be a very lengthy process (e.g., up to a week of processing time for large topological datasets derived from TIGER/Line Files [27]). This would be impractical for large enterprise datasets.

Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.

4.4 Topology Errors and Exceptions

A *topology error* is generated for each instance of a topology rule that is determined to be invalid during the validation process (an example is shown in Fig. 9). Topology rules are commonly specified as a required topological relationship that must hold between collections of features in one or more feature classes. Topology errors are associated with the topology; inspection of the error will enable a user to determine why the error was generated. Topology errors have an associated geometry that is used to position the error in the topology dataset. Topology errors are persisted with the topology.

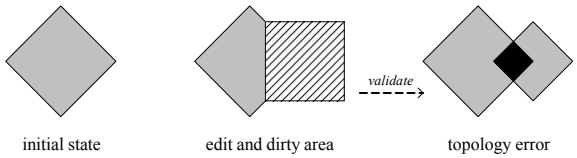


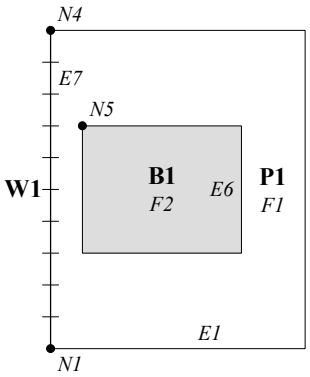
Fig. 9. Example of an edit (polygon creation) followed by a validation where the topology rule is "polygons must not overlap". The generated topology error is depicted by the black diamond

We have observed with large enterprise topologies that it is sometimes the case that certain topology error instances are acceptable. For example, within the Census Bureau's TIGER/db [4] DB system, there is a general rule that American Indian Reservations may not overlap. There are however two instances where this general rule is purposefully violated (e.g., the Lualualei and Mashantucket Pequot reservations in Hawaii, and the Spokane and Schaghticoke reservations in Washington). In order to support such topology error instances where the topology rules are purposefully violated, we add an attribute to a topology error that indicates whether or not the topology error is actually an exception (to the rule). Marking purposeful topology errors as exceptions allows other clients of the topology to handle them in appropriate manners (e.g., report generators that indicate the number of errors in the topology – this is often used as a quantitative measure of quality within the topological dataset). Note that it should also be possible to demote an exception to error status.

4.5 Complexity of Updates

Simple edits to the Geodatabase topological data model can result in rather simple changes to the associated topology metadata (e.g., dirty areas). Consider a simple edit to the sample dataset of Fig. 7 where the building is moved one unit to the right. The changes necessary in the data model are shown in Fig. 10. More specifically, the geometry of the building feature is updates and the dirty area extent of the topology is unioned with the initial and final geometry of the building that was updated (i.e., the polygon (0,3), (0,7),(6,7),(6,3),(0,3)). No other modifications to the persisted representation are necessary.

Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.



Parcels

ID	*Vertices*
P1	(0,0),(0,3),(0,7),(0,10),(8,10),(8,0),(0,0)

Buildings

ID	*Vertices*
B1	*(1,3),(1,7),(6,7),(6,3),(1,3)*

Walls

ID	*Vertices*
W1	(0,10),(0,7),(0,3),(0,0)

Topology Rules

ID	*Rule type*
R1	Parcels no overlap
R2	Buildings no overlap

Topology Errors

ID	*Vertices*	*Rule type*	*Feature*	*IsException*
E1	...	-	-	-

Dirty Area

Vertices
..., (0,3),(0,7),(6,7),(6,3),(0,3)

Fig. 10. An example instance of the Geodatabase topology implementation following an update to the building geometry (r.e., Fig. 7). In this example, updated rows are in italics (i.e., the geometry of the building and the vertices of the dirty area)

5 Implementation Experience

This new topology model has been implemented and is currently shipping with ESRI's ArcGIS 8.3 product. It is fully supported within a multi-user versioned database environment. It has been used to validate very large topologies, including a dataset derived from the features contained within the entire collection of the Census Bureau's TIGER/Line files (53.5 million features; 30 million lines, 23.5 million polygons [18]). A small portion of this dataset in the vicinity of the U.S. Capitol in Washington, DC is shown in Fig. 11. Other large datasets that have been validated are summarized in Table 1.

Table 1. Summary statistics of large topologies validated

Dataset	*feature count*	*topology rule count*
TIGER (U.S. National)	53.5 million	83
LPI, New South Wales, Australia	22.6 million	41
Cook County Assessor (Illinois)	4.3 million	16
Calgary Legal Cadastre (Canada)	2.1 million	5

Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.

Fig. 11. Example of the TIGER dataset taken from the vicinity of the U. S. Capitol in Washington, DC

6 Conclusion

In this paper we described the logical model of GIS topology. We considered a common physical database implementation using the conventional notions for mapping entities and relationships to tables and the conventional primary key / foreign key referential integrity model. Problems in this approach were discussed. We then presented an alternative implementation of the topology model which used an unconventional approach to the problem of database integrity and transaction management.

The presented design serves as the basis for our implementation of topology in the ArcGIS geographic information system. This new models offers increased flexibility in terms of defining which rules constitute a valid topology. This topology model is supported in a multi-user long transaction (i.e., versioned) environment where multiple users may simultaneously edit the same geographic area within the topology extent. This new model has been implemented and is currently shipping with the ArcGIS 8.3 product.

Our future work will focus on experimentation with dirty area management policies that are more useful for the client (i.e., require less revalidation) without incurring considerable

Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.

computational expense during version reconcile, applying the dirty area model to effectively support partial processing in other computationally intensive areas of GIS, supporting this topology model in the distributed database environment, as well as other performance enhancements.

Acknowledgements. Numerous other individuals within ESRI Development were involved in the design and implementation of this topology model in the ArcGIS 8.3 product. Key developers included Jeff Jackson, Jim Tenbrink, and Jan van Roessel. Other large contributors to this work included Craig Gillgrass, Wayne Hewitt, Andy MacDonald, Doug Morganthaler, Jeff Shaner, and François Valois.

References

1. P. Alexandroff. *Elementary Concepts of Topology*. Dover Publications, New York, 1961.
2. B. Baumgart. *Winged-edge Polyhedron Representation*. Technical Report STAN-CS-320, Computer Science Department, Stanford University, Stanford, CA, 1972.
3. G. Boudriault. Topology in the TIGER File. In *Proceedings of the 8th International Symposium on Computer Assisted Cartography (Auto-Carto 8)*, Baltimore, MD, 1987.
4. F. Broome and D. Meixler. The TIGER Data Base Structure. *Cartography and Geographic Information Systems*, 17 (1), January 1990.
5. S. Ceri, R. Cochrane, and J. Widom. Practical Applications of Triggers and Constraints: Successes and Lingering Issues. In *Proceedings of the 26th International Conference on Very Large Data Bases (VLDB'00)*, Cairo, Egypt, 2000.
6. E. Clementini, P. Di Felice, and P. van Oosterom. A Small Set of Formal Topological Relationships Suitable for End-User Interaction. In *Proceedings of the Third International Symposium on Large Spatial Databases (SSD'93)*, Singapore, June, 1993.
7. D. Cooke, and W. Maxfield. The Development of a Geographic Base File and its Uses for Mapping. In *Urban and Regional Information Systems for Social Programs: Papers from the Fifth Annual Conference of the Urban and Regional Information Systems Association*. Kent, OH: Center for Urban Regionalism, Kent State University, 1967.
8. J. Corbett. *Topological Principles in Cartography*. Technical Paper 48. Bureau of the Census, Washington, DC, 1979.
9. M. Egenhofer and J. Herring. *Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases*. Technical Report, Department of Surveying Engineering, University of Maine, 1990.
10. ESRI. *Building a Geodatabase*. Prepared by Environmental Systems Research Institute, ESRI Press, Redlands, CA, 2002.
11. P. Gulutzan and T. Pelzer. *SQL-99 Complete, Really*. Miller Freeman, Lawrence, Kansas, 1999.
12. R. Güting. Realm-Based Spatial Data Types: The ROSE Algebra. *VLDB Journal*, 4 (2), 1995.
13. M. Hammer and D. McLeod. Semantic Integrity in a Relational Database System. In *Proceedings of the 1st International Conference on Very Large Data Bases (VLDB'75)*, Framingham, Massachusetts, September 1975.
14. J. Herring. TIGRIS: Topologically Integrated Geographic Information System. In *Proceedings of the 8th International Symposium on Computer Assisted Cartography (Auto-Carto 8)*, Baltimore, MD, 1987.
15. Intergraph Corp. *GIS: The MGE Way*. Intergraph Technical Paper, October 1995.
16. ISO TC 211/WG 2. *Geographic Information – Spatial Schema. Technical Report, Second Draft of ISO 19107*, International Organization for Standardization, 1999.
17. W. Kim, editor. *Modern Database Systems: The Object Model, Interoperability, and Beyond*. ACM Press, New York, 1995.
18. C. Kinnear. The TIGER Structure. In *Proceedings of the 8th International Symposium on Computer Assisted Cartography (Auto-Carto 8)*, Baltimore, MD, 1987.

Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.

19. V. Markowitz. Safe Referential Integrity Structures in Relational Databases. In *Proceedings of the 17th International Conference on Very Large Data Bases (VLDB'91)*, Barcelona, September 1991.
20. W. May and B. Ludäscher. Understanding the Global Semantics of Referential Actions using Logic Rules. *ACM Transactions on Database Systems*, 27 (4), December 2002.
21. V. Milenkovic. Verifiable Implementations of Geometric Algorithms Using Finite Precision Arithmetic. In *Geometric Reasoning*, The MIT Press, Cambridge, Massachusetts, 1989.
22. S. Morehouse. ARC/INFO: A Geo-Relational Model for Spatial Information. In *Proceedings of the 7th International Symposium on Computer Assisted Cartography (Auto-Carto 7)*, Washington, DC, March 1985.
23. J. Munkres. *Topology*. Second Edition, Prentice-Hall, Englewood Cliffs, New Jersey, 2000.
24. T. Peucker and N. Chrisman. Cartographic Data Structures. In *The American Cartographer*, 2 (2), April 1975.
25. P. Rigaux, M. Scholl, and A. Voisard. *Spatial Databases with Application to GIS*. Morgan Kaufmann, San Francisco, 2002.
26. E. Simon and A. Kotz-Dittrich. Promises and Realities of Active Database Systems. In *Proceedings of the 21st International Conference on Very Large Data Bases (VLDB'95)*, Zürich, Switzerland, 1995.
27. U. S. Census Bureau. *2002 TIGER/Line Technical Documentation*. Prepared by the U. S. Census Bureau, Washington, DC, 2002.
28. P. van Oosterom, J. Stoter, W. Quak, and S. Zlantanova. The Balance Between Geometry and Topology. In *Proceedings of the 2002 Symposium on Spatial Data Handling (SDH'02)*, Ottawa, Canada, July 2002.
29. J. van Roessel. A New Approach to Plane-Sweep Overlay: Topological Structuring and Line-Segment Classification. *Cartography and Geographic Information Systems*, 18 (1), 1991.
30. J. van Roessel. Supporting Multi-Layer Map Overlay and Shared Geometry Management in a GIS. In *Computational Cartography – Cartography Meets Computational Geometry, Dagstuhl Seminar Report 252*, M. Molenaar, M. van Krefeld, and R. Weibel, editors, September 1999.
31. P. Watson. *Topology and ORDBMS Technology*. Laser-Scan White Paper, Laser-Scan Ltd., January 2002.
32. M. Zeiler. *Modeling Our World: The ESRI Guide to Geodatabase Design*. ESRI Press, Redlands, CA, 1999.

Accepted for publication at the *8th International Symposium on Spatial and Temporal Databases (SSTD'03)* – Santorini, Greece, July 2003.