ISO/IEC JTC 1/SC 32/WG 4: TXL-016r2
INCITS H2-2005-107r2
April 21, 2005
Authoritative file: txl016-Topo-Concepts-The-Sequel.pdf
Number of Pages: 18

ISO

International Organization for Standardization

ISO/IEC JTC 1/SC 32

Data Management and Interchange

WG 4

SQL Multimedia and Application Packages

Title: Topo-Geo and Topo-Net 2: Concepts, The Sequel
Author: Paul Scarponcini
Status: SQL/MM—Part 3: Spatial Change Proposal
Source: US Contribution
Abstract:

References:

[1] ISO/IEC CD 13249-3:200x(E) – Text for Continuation CD Editing Meeting **Information technology – Database languages – SQL Multimedia and Application Packages — Part 3: Spatial**, January 24, 2005.

[2] ISO TC211 19107, **Geographic Information – Spatial Schema**, 2001.

[3] ANSI INCITS H2-2003-250, **Approaching Topology**, Paul Scarponcini, April 15, 2003.

[4] ISO TC204 14825, **Intelligent Transport Systems — Geographic Data Files — Overall Data Specification**, October 10, 2002.

[5] ANSI INCITS H2-2003-336, ISO/IEC JTC 1/SC 32/WG 4:MEL-016, **Topology in Three Part Harmony**, Paul Scarponcini, August 8, 2003.

[6] ISO/IEC IS 9075-2:1999, **Information technology – Database languages – SQL – Part 2: Foundation (SQL/Foundation)**, December, 1999.

[7] ISO/IEC JTC 1/SC 32/WG 4:STX-012r1, **Consolidated Ballot Comments on SQL/MM Part 3: Spatial (3rd Edition)**, October, 2004.

[8] ISO/IEC JTC 1/SC 32/WG 4:STX-021r1, **Topo-Geo and Topo-Net 1 – The Concepts**, November, 2004.

1 Introduction

1.1 Ballot Comments addressed

This paper partially addresses ballot comments [7]:

GBR-P03-001	1-Major Technical	*P03-01 Scope*	This clause includes a reference to topology, but there is no support for it in this document. Support for topology should be added. **Solution** None provided with comment.
USA-P03-001	1-Major Technical	*P03-01, Scope*	The first bullet of the second paragraph suggests that spatial data includes topology yet topology is not specified in this part. A new clause for topology needs to be added, consistent with ISO TC211 19107. **Solution** None provided with comment.

It updates the approach proposed for addressing topology by providing revisions to the definitions and concepts relevant to this topic presented in [8] and therefore partially addresses the comments. Additional detail will need to be provided (in subsequent change proposals) in order to completely satisfy the comments.

1.2 Possible Problems addressed

This paper addresses the following possible problem [1]:

3-303 Possible Problem:

Severity: Major Technical
Reference: P03-03.01.02 Definitions Provided in Part 3, P03-04.02 Topology-Geometry, and P03-04.03 Topology-Network
Note at: STX_2004_11_St.Croix_USVI_USA
Source: Paul Scarponcini
Possible Problems in the Definitions and Concepts clauses introduced by ISO/IEC JTC 1/SC 32/WG 4:STX-021r1:
1. A discussion about Face 0 needs to be added.
2. The existing capability of SQL with respect to the generation of unique ID's should be supported.
3. Specification of "optional" values in a column should be re-worded to use proper SQL "NULL" terminology.
4. Functions can only return a single value. Returning row types should be considered as a possible solution.
5. Schema name should be passed as a parameter to all appropriate functions to specify which topology is to be used.
6. "Topologically consistent" should be added to the definitions.
7. Dual sets of functions should be provided where appropriate to enable a choice regarding re-use of unique ID's for splitting and healing functions.
8. A function which returns the ST_Surface value of a face should be added.
9. A function should be added to create a schema and relevant base tables and views (or whatever is to be used to persist the topology data).
10. Replace "storage" with "the site".
11. Add "heal" to the definitions.

This paper partially addresses the following possible problem [1]:

3-304 Possible Problem:

Severity: Major Technical
Reference: P03 (exact location tbd)
Note at: STX_2004_11_St.Croix_USVI_USA
Source: Paul Scarponcini
Possible Problems introduced by ISO/IEC JTC 1/SC 32/WG 4:STX-021r1:
Specification of Topology support is incomplete, as only definitions and concepts have been added. Further specification is required, including the following considerations:
1. GML support needs to be included.
2. The tables in ISO/IEC JTC 1/SC 32/WG 4:STX-021r1 should be specified as views on base tables, with a note about table functions or alternative base tables being acceptable as long as the views are supported.
3. All geometries for a topology schema need to use the same SRS.
4. ID's should be defined as distinct types instead of integers.

1.3 Changes from Previous Version

The following changes were made from the previous version of this paper:

Definitions of heal and split have been revised as per WG4 action.

Definition of topological complex added as from 19107.

"Same topology" changed to "topology".

Description added that <topology-name> and <network-name> are schemas.

1.4 Summary of Approach

Various other documents show the evolution of the proposed topology approach [3, 5, 8]. A tabular approach consistent with the Mini-Topo model of Minimally Redundant Topology [2] has been agreed. Two models are included: Topology-Geometry (Topo-Geo), for spatial topology, includes nodes, edges, and faces whilst Topology-Network (Topo-Net), for linear topology, includes nodes and links. The approach attempts to be consistent with ISO 19107 – Spatial Schema [2] fromTC211, the OpenGIS Consortium, and ISO TC204, Intelligent Transportation Systems, the latter in support of the development of a proposed SQL implementation of their Graphic Data Files (GDF) standard, ISO 14825 [4]. Worked examples were provided in [8].

The eleven points raised in possible problem 3-303 are addressed in this paper as follows:
1. A discussion about Face 0 was added at the end of clause 4.2.3.
2. Support for the existing capability of SQL with respect to the generation of unique ID's is already assumed in this concepts section so no additional action was taken.
3. Specification of "optional" values in a column were re-worded to "possibly nullable" and "mandatory" to "known not nullable." in accordance with proper SQL terminology.
4. The ST_GetFaceEdges function has been changed to return a table containing the face IDs. The ST_ValidateTopoGeo, ST_ValidLogicalNet, and ST_ValidSpatialNet functions have been changed to return a table containing topological inconsistencies. For all other cases where a function returned multiple values (i.e., ST_SplitEdge, ST_AddEdge, ST_SplitLogLink, and ST_SplitGeomLink), the functions have been changed to return only a single value. A subsequent query can be issued to return the additional values.
5. Schema name is already passed as a parameter to all appropriate functions to specify which topology / network is to be used, so no further action has been taken.
6. "Topologically consistent" was added to the definitions.
7. In order to provide dual sets of functions to enable a choice regarding re-use of unique ID's for splitting and healing functions, existing functions which create new ID's are renamed to include "New" in their

name and additional parallel functions were added which re-use an ID and contain “Mod” in their names. ST_SplitEdge becomes ST_NewEdgesSplit and ST_ModEdgeSplit is added; ST_HealEdges becomes ST_NewEdgeHeal and ST_ModEdgeHeal is added; ST_AddEdge becomes ST_AddEdgeNewFaces and ST_AddEdgeModFace is added; ST_RemoveEdge becomes ST_RemEdgeNewFace and ST_RemEdgeModFace is added; ST_SplitLogLink becomes ST_NewLogLinkSplit and ST_ModLogLinkSplit is added; ST_SplitGeomLink becomes ST_NewGeoLinkSplit and ST_ModGeoLinkSplit is added; and ST_HealLinks becomes ST_NewLinkHeal with ST_ModLinkHeal being added.
8. A function which returns the ST_Surface value of a face called ST_GetFaceGeometry has been added to Clause 4.2.3.1.
9. Functions called ST_InitTopoGeo and ST_InitTopoNet have been added to Clauses 4.2.3.2 and 4.3.2.2, respectively, to create schema and relevant base tables and views for the topology-geometry and topology-network data, respectively.
10. The phrase “provides storage for” was replaced by “contains”.
11. The terms “heal” and “split” were added to the definitions.

The following two points raised in possible problem 3-304 were also addressed:
2. For all tables, “table” has been changed to “view”.
3. All geometries for a topology schema are specified as having the same SRS.

The following point raised in possible problem 3-304 will be addressed in a subsequent change proposal:
1. GML support needs to be included.

The author has chosen not to address the following point raised in possible problem 3-304:
4. All integer ID’s have been changed to distinct types
on the basis that, though elegant, it is not essential to the topology solution being proposed.

The following additional changes were made to Topo-Net:
1. Multiple nodes can exist at the same location. This can occur if two roads cross at a grade-separated crossing, and separate nodes are added for the location under and on the bridge.
2. Unlike edges, links have no knowledge about next links or faces. The data about an isolated link is therefore no different than a connected one. Therefore, ST_AddIsoLink and ST_RemoveIsoLink have been generalized into ST_AddLink and ST_RemoveLink, respectively.
3. ST_ChangeLinkGeom has been changed to compare the proposed new geometry start and end points to the link’s start and end node geometries, if they are not null, instead of comparing it to the old geometry.

1.4.1 Completeness

This change proposal only addresses definitions and concepts. Additional change proposal documents have been tabled to complete the topology proposal.

1.5 Conventions

1. SMALLCAPS	denote numbered editorial instructions;
New Text	New text is identified by using a red bold-underlined font except where entirely new sub-clauses are proposed;
~~Deleted Text~~	Deleted text is identified by using a blue strikethrough font;
Plain	denotes existing text to be retained;
Note to reader:...	boxed text enclose notes to the proposal reader;
Note to editor:...	boxed text enclose helpful hints to the editor;

This following abbreviations are used to reference unnumbered sections: "PS" for "Purpose", "DEFN" for "Definition", "DR" for "Definitional Rules", and "DS" for "Description".

1.6 Electronic availability

This paper is available electronically in PDF (.pdf). To access it using anonymous FTP protocols, connect to the following Internet URL:

ftp://sqlstandards.org/SC32/WG4/Meetings/TXL_2005_04_Berlin_DEU

Use the following filename to get the paper:

txl016r1-Topo-Concepts-The-Sequel.pdf*PDF*

2 Proposal

2.1 Add the following new subclauses to subclause 3.1.2 “Definitions provided in Part 3”:

3.1.2.24+1
heal
topological operation which amalgamates two edges (or links) into one by deleting their common node or which amalgamates two faces into one by deleting their common edge

NOTE Heal is the reverse operation of split. When healing two successive edges (or links), the connecting node is deleted and a single edge (or link) replaces the original two. When healing two adjacent faces, their common edge is deleted and a single face replaces the original two.

3.1.2.40+1
split
topological operation which creates two edges (or links) from one by inserting a node or which creates two faces from one by inserting an edge.

NOTE Split is the reverse operation of split.

3.1.2.41+1
topologically consistent
a topology which exhibits the following characteristics:

1) all topological complexes are fully decomposed into their topological primitives
2) no two nodes exist at the same position in space
3) a node exists at the beginning and end of every edge
4) no edge has a geometry which crosses the geometry of a node
5) no edge has a geometry which crosses, overlaps, or is contained within the geometry of another edge
6) all edge geometries are simple
7) all edge geometries have a start point equal to the geometry of their start node
8) all edge geometries have a end point equal to the geometry of their end node
9) no face has a geometry which overlaps the geometry of another face
10) no face has a geometry within the geometry of another face
11) all geometries for the topology have the same spatial reference system.

2.2 Add the following new subclause to subclause 3.1.4 “Definitions taken from ISO 19107”:

af+1) topological complex

2.3 Changes made to Clause 4.2 Topology-Geometry:

1. CHANGE CLAUSE 4.2 TOPOLOGY-GEOMETRY AS FOLLOWS:

The following ~~tables~~ **views** are defined: <topology-name>.ST_NODE, <topology-name>.ST_EDGE, and <topology-name>.ST_FACE**, where <topology-name> is an SQL-Schema**.

The rows in these ~~tables~~ **views** define topological primitives of type node, edge, and face, respectively for the Topology-Geometry (ST_Topo-Geo) called <topology-name>. Nodes and edges have associated ST_Geometries. Faces may have a minimum bounding rectangle geometry for spatial indexing. **All geometry values for a given ST_Topo-Geo shall have the same spatial reference system.**

Each topological primitive has an ID, unique within the respective ~~table~~ **view**, which allows the primitive to be referenced from another (e.g., feature~~,~~ **or** ST_Topo-Geo~~, or ST_Topo-Net~~) ~~table~~ **view**.

4.2.1 <topology-name>.ST_NODE

The <topology-name>.ST_NODE ~~table~~ **view** ~~provides storage for~~ **contains** the node type of topological primitives (ST_Node) contained in the <topology-name> Topology-Geometry. An ST_Node has a ~~mandatory~~ **known not nullable**, unique node ID of type integer and a **known not nullable** geometry of type ST_Point. If ST_Node is an isolated node, it has a containing face, identified by a~~n~~ <topology-name>.ST_FACE.ID.

4.2.1.1 Routines on <topology-name>.ST_NODE only

1) ST_AddIsoNode: for the provided topology-name, optional face ID, and ST_Point geometry, inserts a row into the <topology-name>.ST_NODE ~~table~~ **view corresponding to an isolated node**, returning the automatically generated, unique integer node ID. If a face ID is provided, the ST_Point geometry must be within the geometry of the face or an exception is raised. If another node in the <topology-name>.ST_NODE ~~table~~ **view** exists at the ST_Point location **or if the geometry of an existing edge crosses the ST_Point location**, an exception is raised.
2) ST_MoveIsoNode: for the provided topology-name, node ID, and ST_Point geometry, updates the ST_Point geometry value. If the node is a connected node or if another node in the <topology-name>.ST_NODE ~~table~~ **view** exists at the new location **or if the geometry of an existing edge crosses the ST_Point location**, an exception is raised.
3) ST_RemoveIsoNode: deletes the row for the isolated node identified by the provided topology-name and node ID. If the node is a connected node, an exception is raised.

4.2.2 <topology-name>.ST_EDGE

The <topology-name>.ST_EDGE ~~table~~ **view** ~~provides storage for~~ **contains** the edge type of topological primitives (ST_Edge) contained in the <topology-name> Topology-Geometry. An ST_Edge has a ~~mandatory~~ unique edge ID of type integer; ~~mandatory~~ node ID's of type integer for the start and end nodes; ~~mandatory~~ edge ID's of type integer for the next left face and next right face edges; ~~mandatory~~ face ID's of type integer for the left and right faces; and a ~~mandatory~~ geometry of type ST_Curve. **All values are known not nullable.** An isolated edge will have its containing face as both its left and right faces. The next right face edge will have an ID equal to the edge ID of the isolated edge and the next left face edge will have the negative of the isolated edge's ID.

Start and end node ID's are immutable. To change the start or end node of an edge, the edge shall be removed and a new one shall be created.

4.2.2.1 Routines on <topology-name>.ST_EDGE only

1) ST_AddIsoEdge: for the provided topology-name, start and end node IDs, and ST_Curve geometry, inserts a row into the <topology-name>.ST_EDGE ~~table~~ **view**, returning the automatically generated, unique integer edge ID. The next right face ~~ID~~ **edge** is set equal to the new edge ID and the next left face edge is set equal to the negative of this value. The left and right face IDs are set equal to the containing face ID of the start and end node.

An exception is raised for any of the following conditions:

a) ~~if the start and end nodes are not distinct, isolated nodes in the <topology-name>.ST_NODE table,~~**if the start and end node IDs correspond to existing isolated nodes,**

b) if the start and end node containing face IDs are not equal,

c) if the ST_Point geometry of the start node does not equal the ST_StartPoint of the ST_Curve geometry of the edge,

d) if the ST_Point geometry of the end node does not equal the ST_EndPoint of the ST_Curve geometry of the edge,

e) if the ST_Curve geometry is not within the geometry of the containing face of the start and end nodes,

f) if the ST_Curve geometry intersects the ST_Point geometry of any isolated node other than the start and end node, or

g) if the ST_Curve geometry intersects the ST_Curve geometry of any other edge in the <topology-name>.ST_EDGE ~~table~~ **view**.

2) ST_GetFaceEdges: for the provided topology-name and face ID, returns **a table containing** the integer edge IDs ~~of~~ **for** the edges which bound the face, in counterclockwise order. Edge IDs will be negated in the query result if the face is right of the edge when looking in the direction of the edge from start to end node.

3) ST_ChangeEdgeGeom: for the provided topology-name, edge ID, and ST_Curve geometry, updates the ST_Curve geometry value.

An exception is raised for any of the following conditions:

a) if the ST_StartPoint of the new ST_Curve geometry is not equal to the ST_StartPoint of the existing ST_Curve geometry,

b) if the ST_EndPoint of the new ST_Curve geometry is not equal to the ST_EndPoint of the existing ST_Curve geometry,

c) if the interior of the new ST_Curve geometry intersects the ST_Point geometry of any isolated node in the <topology-name>.ST_NODE ~~table~~ **view**, or

d) if the interior of the new ST_Curve geometry intersects the ST_Curve geometry of any other edge in the <topology-name>.ST_EDGE ~~table~~ **view**.

4) ST_RemoveIsoEdge: deletes the row for the isolated edge identified by the provided topology-name and edge ID. The start and end nodes are not removed from the <topology-name>.ST_NODE ~~table~~ **view**. If the edge is a not an isolated edge, an exception is raised.

4.2.2.2 Routines on <topology-name>.ST_NODE and <topology-name>.ST_EDGE

1) ST_~~SplitEdge~~**NewEdgesSplit**: **splits an edge by creating a new node along an existing edge, deleting the original edge and replacing it with two new edges. For** ~~for~~ the provided topology-name, edge ID, and ST_Point geometry,

a) inserts a row into the <topology-name>.ST_NODE ~~table~~ **view**, with geometry equal to the input ST_Point value,

b) returns the automatically generated, unique integer node ID,

c) deletes the row in the <topology-name>.ST_EDGE ~~table~~ **view** for the edge identified by the provided topology-name and edge ID, and

d) inserts two rows into the <topology-name>.ST_EDGE ~~table~~ **view** for the two new resultant edges, deriving appropriate node, edge, and face values from the deleted edge,

e) creates ST_Curve geometries for the two new edges by splitting the geometry of the split edge at the ST_Point location,

f) ~~returns the two automatically generated, unique integer edge IDs, and~~

~~g)~~ makes any necessary updates to the next left and right face edge IDs for any edges incident on the start and end nodes of the edge being split.

To determine the two new edge IDs, query the <topology-name>.ST_EDGE view for edges with a start or end node equal to the returned node ID. Both new edges have the same direction as the edge being split. An exception is raised for any of the following conditions:

a) if the edge identified by the edge ID does not exist in the <topology-name>.ST_EDGE ~~table~~ **view**,

b) if the ST_Point geometry is not within the ST_Curve geometry of the identified edge, or

c) if a node already exists in the <topology-name>.ST_NODE ~~table~~ **view** at the input ST_Point geometry location.

2) ST_ModEdgeSplit: splits an edge by creating a new node along an existing edge, modifying the original edge and adding a new edge. For the provided topology-name, edge ID, and ST_Point geometry,

a) inserts a row into the <topology-name>.ST_NODE view, with geometry equal to the input ST_Point value,

b) returns the automatically generated, unique integer node ID,

c) modifies the row in the <topology-name>.ST_EDGE view for the edge identified by the provided topology-name and edge ID, deriving appropriate node, edge, and face values from the original edge and new node,

d) inserts a new row into the <topology-name>.ST_EDGE view for the other new resultant edge, deriving appropriate node, edge, and face values from the original edge and new node,

e) creates ST_Curve geometries for the new and modified edges by splitting the geometry of the original edge at the ST_Point location,

f) makes any necessary updates to the next left and right face edge IDs for any edges incident on the start and end nodes of the edge being split.

To determine the new edge ID, query the <topology-name>.ST_EDGE view for edges with a start or end node equal to the returned node ID. The new and modified edges have the same direction as the original edge. An exception is raised for any of the following conditions:

a) if the edge identified by the edge ID does not exist in the <topology-name>.ST_EDGE view,

b) if the ST_Point geometry is not within the ST_Curve geometry of the identified edge, or

c) if a node already exists in the <topology-name>.ST_NODE view at the input ST_Point geometry location.

~~2~~**3**) ST_~~HealEdges~~**NewEdgeHeal**: **heals two edges by deleting the node connecting them, deleting both edges, and replacing them with a new edge whose direction is the same as the first edge provided. For** ~~for~~ the provided topology-name and two edge IDs,

a) deletes the row in the <topology-name>.ST_NODE ~~table~~ **view** corresponding to the node shared by the two identified edges,

b) deletes the two rows in the <topology-name>.ST_EDGE ~~table~~ **view** identified by the input edge IDs,

c) inserts a new row into the <topology-name>.ST_EDGE ~~table~~ **view** for the resultant edge, deriving appropriate node, edge, and face values from the deleted edges,

d) creates an ST_Curve geometry for the new edge from the geometries of the two deleted edges,

e) returns the automatically generated, unique integer edge ID for the new edge, and

f) makes any necessary updates to the next left and right face edge IDs for any edges incident on the start and end nodes of the edges being healed.

The direction of the new edge shall be the same as the direction of the first supplied edge. An exception is raised for any of the following conditions:

a) if either edge identified by the edge IDs does not exist in the <topology-name>.ST_EDGE ~~table~~ **view**,

b) if the two edges do not share a common node, or

c) if additional edges also share the common node.

4) ST_ModEdgeHeal: heals two edges by deleting the node connecting them, modifying the first edge provided, and deleting the second edge. For the provided topology-name and two edge IDs,

a) deletes the row in the <topology-name>.ST_NODE view corresponding to the node shared by the two identified edges,

b) deletes the row in the <topology-name>.ST_EDGE view identified by the second input edge ID,

c) modifies the values in the row in the <topology-name>.ST_EDGE view for the other edge, deriving appropriate node, edge, and face values from the original edges,

d) creates an ST_Curve geometry for modified edge from the geometries of the two original edges,

e) makes any necessary updates to the next left and right face edge IDs for any edges incident on the start and end nodes of the edges being healed.

An exception is raised for any of the following conditions:

a) if either edge identified by the edge IDs does not exist in the <topology-name>.ST_EDGE view,

b) if the two edges do not share a common node, or

c) if additional edges also share the common node.

4.2.3 <topology-name>.ST_FACE

The <topology-name>.ST_FACE ~~table~~ **view** ~~provides storage for~~ **contains** the face type of topological primitives (ST_Face) contained in the <topology-name> Topology-Geometry. An ST_Face has a ~~mandatory~~ **known not nullable**, unique face ID of type integer and ~~an optional~~ **a possibly nullable** MBR (minimum bounding rectangle) geometry of type ST_Polygon.

The <topology-name>.ST_FACE view contains a row for the universal face. The universal face contains everything else in the topology exterior to all other faces. This face has a face ID = 0 (zero). There is no geometry associated with the universal face.

4.2.3.1 Routines on <topology-name>.ST_EDGE and <topology-name>.ST_FACE

1) ST_AddEdge**NewFaces**: **adds a new edge and, if in doing so it splits a face, deletes the original face and replaces it with two new faces. For** ~~for~~ the provided topology-name, start and end node IDs, and ST_Curve geometry,

a) inserts a row into the <topology-name>.ST_EDGE ~~table~~ **view**, with start and end nodes as specified, automatically determined next edges and left and right faces, and geometry equal to the input ST_Curve value,

b) returns the automatically generated, unique integer edge ID, and

c) if the new edge splits a face, then

i) deletes the row in the <topology-name>.ST_FACE ~~table~~ **view** corresponding to the face being split,

ii) automatically generates two new unique integer face IDs,

iii) inserts two rows into the <topology-name>.ST_FACE ~~table~~ **view** for the two new resultant faces,

iv) creates ST_Polygon MBR geometries for the two new faces, **and**

~~iv) returns the two automatically generated, unique integer face IDs, and~~

v) updates the appropriate next left and right face edge IDs and left and right face IDs for edges bounding the face being split.

To determine the two new face IDs, query the <topology-name>.ST_EDGE view for the left and right faces for the edge with the returned edge ID. An exception is raised for any of the following conditions:

a) if either the start or end nodes identified do not exist in the <topology-name>.ST_NODE ~~table~~ **view**,

b) if the ST_StartPoint of the new ST_Curve geometry is not equal to the ST_Point value of the start node geometry,

c) if the ST_EndPoint of the new ST_Curve geometry is not equal to the ST_Point value of the end node geometry,

d) if the interior of the new ST_Curve geometry intersects the ST_Point geometry of any isolated node in the <topology-name>.ST_NODE ~~table~~ **view**,

e) if the interior of the new ST_Curve geometry intersects the ST_Curve geometry of any other edge in the <topology-name>.ST_EDGE ~~table~~ **view**, or

f) if an edge already exists in the <topology-name>.ST_EDGE ~~table~~ **view** with the same terminal nodes and geometry.

2) ST_AddEdgeModFace: adds a new edge and if in doing so it splits a face, modifies the original face and adds a new face. For the provided topology-name, start and end node IDs, and ST_Curve geometry,

a) inserts a row into the <topology-name>.ST_EDGE view, with start and end nodes as specified, automatically determined next edges and left and right faces, and geometry equal to the input ST_Curve value,

b) returns the automatically generated, unique integer edge ID, and

c) if the new edge splits a face, then

i) modifies the ST_Polygon MBR geometry in the <topology-name>.ST_FACE view for the face being split,

ii) inserts a new row into the <topology-name>.ST_FACE view for the other new resultant face,

iii) creates an ST_Polygon MBR geometry for the new face, and

iv) updates the appropriate next left and right face edge IDs and left and right face IDs for edges bounding the face being split.

To determine the new and modified face IDs, query the <topology-name>.ST_EDGE view for the left and right faces for the edge with the returned edge ID. An exception is raised for any of the following conditions:

a) if either the start or end nodes identified do not exist in the <topology-name>.ST_NODE view,

b) if the ST_StartPoint of the new ST_Curve geometry is not equal to the ST_Point value of the start node geometry,

c) if the ST_EndPoint of the new ST_Curve geometry is not equal to the ST_Point value of the end node geometry,

d) if the interior of the new ST_Curve geometry intersects the ST_Point geometry of any isolated node in the <topology-name>.ST_NODE view,

e) if the interior of the new ST_Curve geometry intersects the ST_Curve geometry of any other edge in the <topology-name>.ST_EDGE view, or

f) if an edge already exists in the <topology-name>.ST_EDGE view with the same terminal nodes and geometry.

~~2~~3) ST_Rem~~ove~~Edge**NewFace**: **removes an edge and, if the removed edge separated two faces, deletes the original faces and replaces them with one new face. For** ~~for~~ the provided topology-name and edge ID,

a) deletes the row in the <topology-name>.ST_EDGE ~~table~~ **view** identified by the edge ID, and

b) if the edge removal results in the healing of two faces, then

i) deletes the two rows in the <topology-name>.ST_FACE ~~table~~ **view** corresponding to the faces being healed,

ii) inserts a new row into the <topology-name>.ST_FACE ~~table~~ **view** for the new resultant face,

iii) creates an ST_Polygon MBR geometry for the new face,

iv) returns the automatically generated, unique integer face ID, and

v) updates the appropriate next left and right face edge IDs and left and right face IDs for edges bounding the faces being healed.

The start and end nodes of the deleted edge remain in the <topology-name>.ST_NODE ~~table~~ **view**. An exception is raised if the edge identified by the edge ID does not exist in the <topology-name>.ST_EDGE ~~table~~ **view**.

4) ST_RemEdgeModFace: removes an edge and, if the removed edge separated two faces, heals the two faces by modifying one of the faces and deleting the other. For the provided topology-name and edge ID,

a) deletes the row in the <topology-name>.ST_EDGE view identified by the edge ID, and

b) if the edge removal results in the healing of two faces, then

i) deletes the row in the <topology-name>.ST_FACE view corresponding to one of the faces being healed,

ii) creates a new ST_Polygon MBR geometry for the modified face,

iii) updates the appropriate next left and right face edge IDs and left and right face IDs for edges bounding the faces being healed.

The choice of which face to modify and which to delete is implementation-dependent.

The start and end nodes of the deleted edge remain in the <topology-name>.ST_NODE view. An exception is raised if the edge identified by the edge ID does not exist in the <topology-name>.ST_EDGE view.

5) ST_GetFaceGeometry: for the provided topology-name and face ID, returns the exact geometry of the face:

a) determines the edges in the <topology-name>.ST_Edge view which bound the face identified by the face ID

b) retrieves the ST_Curve geometries from the <topology-name>.ST_Edge view for each of these edges

c) returns an ST_Surface geometry value constructed from the edge geometries.

An exception is raised if the face identified by the face ID does not exist in the <topology-name>.ST_FACE view.

4.2.3.2 Routines on <topology-name>.ST_NODE, <topology-name>.ST_EDGE and <topology-name>.ST_FACE

1) ST_InitTopoGeo: for the provided topology-name, creates the <topology-name> schema and the ST_NODE, ST_EDGE, and ST_FACE views for this schema. An exception is raised if a schema already exists with that name.

~~1~~2) ST_CreateTopoGeo: for the provided topology-name, and ST_GeomCollection, populates the <topology-name>.ST_NODE, <topology-name>.ST_EDGE, and <topology-name>.ST_FACE ~~tables~~ views from the geometry values in the ST_GeomCollection. An exception is raised if any of these three ~~tables~~ views do not already exist or if they already contain any rows.

~~2~~3) ST_ValidateTopoGeo: for the provided topology-name, returns a table containing topological inconsistencies. ~~determines whether the rows in the <topology-name>.ST_NODE, <topology-name>.ST_EDGE, and <topology-name>.ST_FACE tables are topologically consistent.~~

2.4 Changes made to Clause 4.3 Topology-Network:

1. CHANGE CLAUSE 4.3 TOPOLOGY-NETWORK AS FOLLOWS:

The following ~~tables~~ **views** are defined: <network-name>.ST_NETNODE and <network-name>.ST_NETLINK**, where <network-name> is an SQL-Schema**.

The rows in these ~~tables~~ **views** define network primitives of type node and link, respectively for the Topology-Network (ST_Topo-Net) called <network-name>. These network primitives may have associated ST_Geometries. **All geometry values for a given ST_Topo-Net shall have the same spatial reference system.**

Each topological primitive has an ID, unique within the respective ~~table~~ **view**, which allows the primitive to be referenced from another (e.g., feature or ST_Topo-Net) ~~table~~ **view**.

4.3.1 <network-name>.ST_NETNODE

The <network-name>.ST_NETNODE ~~table~~ **view** ~~provides storage for~~ **contains** the node type of network primitives (ST_Node) contained in the <network-name> Topology-Network. An ST_Node has a ~~mandatory~~ **known not nullable**, unique node ID of type integer and ~~an optional~~ **a possibly nullable** geometry of type ST_Point.

4.3.1.1 Routines on <network-name>.ST_NETNODE only

1) ST_AddIsoNode: for the provided network-name, and optional ST_Point geometry, inserts a row into the <network-name>.ST_NETNODE ~~table~~ **view**, returning the automatically generated, unique integer node ID. ~~If another node in the <network-name>.ST_NETNODE table exists at the ST_Point location, an exception is raised.~~

2) ST_MoveIsoNode: for the provided network-name, node ID, and ST_Point geometry, updates the ST_Point geometry value. If the node is a connected node ~~or if another node in the <network-name>.ST_NETNODE table already exists at the ST_Point location~~**, an exception is raised**.

3) ST_RemoveIsoNode: deletes the row for the isolated node identified by the provided network-name and node ID. If the node is a connected node, an exception is raised.

4.3.2 <network-name>.ST_NETLINK

The <network-name>.ST_NETLINK ~~table~~ **view** ~~provides storage for~~ **contains** the link type of network primitives (ST_Link) contained in the <network-name> Topology-Network. An ST_Link has a ~~mandatory~~ **known not nullable** unique link ID of type integer; ~~mandatory~~ **known not nullable** node IDs of type integer for the start and end nodes; and ~~an optional~~ **a possibly nullable** geometry of type ST_Curve.

Start and end node ID's are immutable. To change the start or end node of an link, the link shall be removed and a new one shall be created.

4.3.2.1 Routines on <network-name>.ST_NETLINK only

1) ST_Add~~Iso~~Link: for the provided network-name, start and end node IDs, and optional ST_Curve geometry, inserts a row into the <network-name>.ST_NETLINK ~~table~~ **view**, returning the automatically generated, unique integer link ID.

   An exception is raised for any of the following conditions:

   a) if the start and end nodes are not existing ~~isolated~~ nodes in the <network-name>.ST_NETNODE ~~table~~ **view**,

   b) if an ST_Curve geometry is provided, then

      i) if the start node has a geometry and the location thereby specified does not equal the location of the ST_StartPoint of the proposed ST_Curve geometry of the link,

ii) if the end node has a geometry and the location thereby specified does not equal the location of the ST_EndPoint of the proposed ST_Curve geometry of the link~~, or.~~

2) ST_ChangeLinkGeom: for the provided network-name, link ID, and ST_Curve geometry, updates the ST_Curve geometry value.

An exception is raised if the link identified by the provided link ID has a non-NULL geometry value and

a) if the ST_StartPoint of the new ST_Curve geometry is not equal to the ~~ST_StartPoint of the existing ST_Curve~~ geometry **of the start node of the link**, or

b) if the ST_EndPoint of the new ST_Curve geometry is not equal to the ~~ST_EndPoint of the existing ST_Curve~~ geometry **of the start node of the link.**~~,~~

3) ST_Remove~~Iso~~Link: deletes the row for the ~~isolated~~ link identified by the provided network-name and link ID. The start and end nodes are not removed from the <network-name>.ST_NETNODE ~~table~~ **view**. ~~If the link is a not an isolated link, an exception is raised.~~

4.3.2.2 Routines on <network-name>.ST_NETNODE and <network-name>.ST_NETLINK

1) ST_InitTopoNet: for the provided network-name, creates the <network-name> schema and the ST_NETNODE and ST_NETLINK views for this schema. An exception is raised if a schema already exists with that name.

~~1~~**2**) ST_~~SplitLogLink~~**NewLogLinkSplit**: **splits a link in a logical network by creating a new node along an existing link, deleting the original link and replacing it with two new links. For** ~~for~~ the provided network-name and link ID,

a) inserts a row into the <network-name>.ST_NETNODE ~~table~~ **view**, with NULL values for Geometry,

b) returns the automatically generated, unique integer node ID,

c) deletes the row in the <network-name>.ST_NETLINK ~~table~~ **view** for the link identified by the provided network-name and link ID, **and**

d) inserts two rows into the <network-name>.ST_NETLINK ~~table~~ **view** for the two new resultant links, deriving appropriate start and end node IDs from the deleted link and newly generated node~~, and~~

~~e) returns the two automatically generated, unique integer link IDs~~.

To determine the two new link IDs, query the <network-name>.ST_NETLINK view for links with a start or end node equal to the returned node ID. Both new links have the same direction as the link being split. An exception is raised if the link identified by the link ID does not exist in the <network-name>.ST_NETLINK ~~table~~ **view**,

3) ST_ModLogLinkSplit: splits a logical network link by creating a new node along an existing link, modifying the original link and adding a new link. For the provided network-name and link ID,

a) inserts a row into the <network-name>.ST_NETNODE view, with NULL values for Geometry,

b) returns the automatically generated, unique integer node ID,

c) modifies the row in the <network-name>.ST_NETLINK view for the link identified by the provided network-name and link ID, deriving appropriate start and end node IDs from the original link and newly generated node, and

d) inserts a new row into the <network-name>.ST_NETLINK view for the other new resultant link, deriving appropriate start and end node IDs from the original link and newly generated node.

To determine the new link ID, query the <network-name>.ST_NETLINK view for links with a start or end node equal to the returned node ID. The new and modified links have the same direction as the link being split. An exception is raised if the link identified by the link ID does not exist in the <network-name>.ST_NETLINK view,

2**4**) ST_SplitGeomLink**NewGeoLinkSplit**: **splits a link in a network with geometry by creating a new node along an existing link, deleting the original link and replacing it with two new links. For** ~~for~~ the provided network-name, link ID, and ST_Point geometry,

a) inserts a row into the <network-name>.ST_NETNODE ~~table~~, with geometry equal to the input ST_Point value,

b) returns the automatically generated, unique integer node ID,

c) deletes the row in the <network-name>.ST_NETLINK ~~table~~ for the link identified by the provided network-name and link ID,

d) inserts two rows into the <network-name>.ST_NETLINK ~~table~~ for the two new resultant links, deriving appropriate start and end node IDs from the deleted link and newly generated node, **and**

e) creates ST_Curve geometries for the two new links by splitting the geometry of the split link at the ST_Point location~~, and~~

~~f) returns the two automatically generated, unique integer link IDs~~.

To determine the two new link IDs, query the <network-name>.ST_NETLINK view for links with a start or end node equal to the returned node ID. Both new links have the same direction as the link being split. An exception is raised for any of the following conditions:

a) if the link identified by the link ID does not exist in the <network-name>.ST_NETLINK ~~table~~,

b) if the ST_Point geometry is not within the ST_Curve geometry of the identified link,

c) ~~if a node already exists in the <network-name>.ST_NETNODE table at the input ST_Point geometry location, or~~

~~d)~~ if the link identified by the link ID contains a null geometry value.

5) ST_ ModGeoLinkSplit: splits a link in a network with geometry by creating a new node along an existing link, modifying the original link and adding a new link. For the provided network-name and link ID,

a) inserts a row into the <network-name>.ST_NETNODE view, with geometry equal to the input ST_Point value,

b) returns the automatically generated, unique integer node ID,

c) modifies the row in the <network-name>.ST_NETLINK view for the link identified by the provided network-name and link ID, deriving appropriate start and end node IDs from the original link and newly generated node,

d) inserts a new row into the <network-name>.ST_NETLINK view for the other new resultant link, deriving appropriate start and end node IDs from the original link and newly generated node, and

e) creates ST_Curve geometries for the new and modified links by splitting the geometry of the split link at the ST_Point location.

To determine the new link ID, query the <network-name>.ST_ NETLINK view for links with a start or end node equal to the returned node ID. The new and modified links have the same direction as the link being split.

An exception is raised for any of the following conditions:

a) if the link identified by the link ID does not exist in the <network-name>.ST_NETLINK table,

b) if the ST_Point geometry is not within the ST_Curve geometry of the identified link,

c) if the link identified by the link ID contains a null geometry value.

~~3~~6)ST_~~HealLinks~~NewLinkHeal: heals two links by deleting the node connecting them, deleting both links, and replacing them with a new link. For ~~for~~ the provided network-name and two link IDs,

a) if no other links start or end at the node shared by the two identified links, deletes the row in the <network-name>.ST_NETNODE ~~table~~ view corresponding to the node shared by the two identified links,

b) deletes the two rows in the <network-name>.ST_NETLINK ~~table~~ view identified by the input link IDs,

c) inserts a new row into the <network-name>.ST_NETLINK ~~table~~ view for the resultant link, deriving appropriate start and end node IDs from the deleted links,

d) if the two links have geometry, creates an ST_Curve geometry for the new link from the geometries of the two deleted links,

e) returns the automatically generated, unique integer link ID for the new link.

The direction of the new link shall be the same as the direction of the first supplied link. An exception is raised for any of the following conditions:

a) if either link identified by the link IDs does not exist in the <network-name>.ST_NETLINK ~~table~~ view, or

b) if the two links do not share a common node.

7) ST_ModLinkHeal: heals two links by deleting the node connecting them, modifying one of the links, and deleting the other. For the provided network-name and two link IDs,

a) if no other links start or end at the node shared by the two identified links, deletes the row in the <network-name>.ST_NETNODE view corresponding to the node shared by the two identified links,

b) deletes the row in the <network-name>.ST_NETLINK view identified by one of the input link IDs,

c) modifies the values in the row in the <network-name>.ST_NETLINK view for the other link, deriving appropriate start and end node IDs from the original links,

d) if the two links had geometry, creates a new ST_Curve geometry for the modified link from the geometries of the two original links.

The choice of which link to delete and which to modify as well as the resultant direction of the modified link are implementation-dependent.

An exception is raised for any of the following conditions:

a) if either link identified by the link IDs does not exist in the <network-name>.ST_NETLINK view, or

b) if the two links do not share a common node.

~~4~~8) ST_LogiNetFromTGeo: for the provided network-name, and topology-name, creates a logical network by populating the <network-name>.ST_NETNODE and <network-name>.ST_NETLINK ~~tables~~ views from the Topo-Geo values identified by the provided topology-name. A Topo-Net node will be created for each Topo-Geo node. A Topo-Net link will be created for each Topo-Geo edge. A logical network will result, with nodes and links having geometry values set to NULL. An exception is raised if either of the two Topo-Net ~~tables~~ views do not already exist or if they already contain any rows or if any of the three Topo-Geo ~~tables~~ views do not exist.

~~5~~**9**) ST_SpatNetFromTGeo: for the provided network-name, and topology-name, creates a ~~logical~~ **spatial** network by populating the <network-name>.ST_NETNODE and <network-name>.ST_NETLINK ~~tables~~ **views** from the Topo-Geo values identified by the provided topology-name. A Topo-Net node will be created for each Topo-Geo node. A Topo-Net link will be created for each Topo-Geo edge. A spatial network will result, with nodes and links having geometry values obtained from their corresponding Topo-Geo primitives. An exception is raised if either of the two Topo-Net ~~tables~~ **views** do not already exist or if they already contain any rows or if any of the three Topo-Geo ~~tables~~ **views** do not exist.

~~6~~**10**) ST_SpatNetFromGeom: for the provided network-name, and ST_GeomCollection, creates a spatial network by populating the <network-name>.ST_NETNODE and <network-name>.ST_NETLINK ~~tables~~ **views** from the geometry values in the ST_GeomCollection. A node will be created wherever links start, end, or cross. A spatial network will result, with nodes and links having geometry values. An exception is raised if either of these two ~~tables~~ **views** do not already exist or if they already contain any rows.

~~7~~**11**) ST_ValidLogicalNet: for the provided network-name, **returns a table containing logical network inconsistencies.**~~determines whether the rows in the <network-name>.ST_NETNODE and <network-name>.ST_NETLINK tables constitute a consistent logical network~~.

~~8~~**12**) ST_ValidSpatialNet: for the provided network-name, **returns a table containing spatial network inconsistencies.**~~determines whether the rows in the <network-name>.ST_NETNODE and <network-name>.ST_NETLINK tables constitute a consistent spatial network with geometries.~~

3 Checklist

Concepts	Yes
Static methods	
Constructor methods	
Cast definitions	
Ordering definitions	
SQL Transforms definitions	
Conformance Clause	
New Status Codes	
Closing Possible Problems	
Any new Possible Problems	
18-Character identifiers	Yes
Full data type names for basetypes	
Implementation-defined elements	
Implementation-defined Meta-variables	
Implementation-dependent elements	
Implementation-dependent Meta-variables	
Deprecated items	No
Incompatibilities	
Part 3: Spatial Only	
ST_Geometry Type Hierarchy	
Well-known Text Support	
Well-known Binary Support	
GML Support	
Empty Set Support	

End of Paper.