

**Title:** Response to INCITS H2-2004-168r2  
**Author:** Mark Ashworth with Krishana Kulkarni  
**Source:** Expert Contribution  
**Abstract:**

INCITS H2-2004-168r2 has made a good start in responding to USA-P03-001. In this paper, we discuss some of the issues that came up while our experts reviewed the functionality in [H2168R2]

**References:**

[H2168R2]      INCITS H2-2004-168r2, Topo-Geo and Topo-Net 1: The Concepts, Paul Scarponcini  
[ISO19107]     ISO 19107, Geographic information — Spatial schema

## **1      Introduction**

[H2168R2] has made a good start in responding to USA-P03-001. In this paper, we discuss some of the issues that came up while our experts reviewed the functionality in [H2168R2].

We note in this revision ([H2168R2]) that there are a list of items identified in the "Other issues raised that have not (yet) been addressed" list on page 2 and we feel they need to be addressed before going forward. This paper includes some further discussion of some of these items along with some additional issues we found during the review of the revision.

### **1.1      Modeling with the Topology-Geometry Model**

This subclause addresses an issue identified in the "Other issues raised that have not (yet) been addressed" on Page 2 of 22 of [H2168r2], namely:

- linking topological primitives to features (currently implementation-defined)

#### **1.1.1      Modeling with the existing Geometry Model**

With the existing Geometry Model in SQL/MM, a user is provided with self-contained values representing the location of some real-world thing. In the "Requirements" in [H2168R2], the examples of real-world thing are:

Park  
Parcel  
Road  
Fountain

With the geometry model, real-world things are typically modeled as tables with a column to represent the Geometry. For example:

```
CREATE TABLE Park (  
    PARK_NAME CHARACTER(16),  
    PARK_GEOMETRY ST_Polygon  
);  
  
CREATE TABLE Parcel (  
    PARCEL_ID CHARACTER(16),  
    PARCEL_GEOMETRY ST_Polygon  
);  
  
CREATE TABLE Road (  
    ROAD_NAME CHARACTER(16),  
    ROAD_GEOMETRY ST_Line  
);
```

```

        ROAD_NAME CHARACTER(20),
        ROAD_GEOMETRY ST_LineString
    );

    CREATE TABLE Fountain (
        FOUNTAIN_NAME CHARACTER(16),
        LOCATION ST_Point
    );

```

To manage the geometry in this model is done by inserting or deleting rows in the table or updating the ST\_Geometry column of such table. Since they are self-contained, it does not affect other values.

Here is an example of a Park 'Park A' and Road 'Road 5' being added to the model.

```

INSERT INTO Park(PARK_NAME, PARK_GEOMETRY)
VALUES ('Park A', ST_PolyFromText('polygon((8 8,20 8, 20 4 8 4, 8
8))'));

INSERT INTO Road(ROAD_NAME, ROAD_GEOMETRY)
VALUES ('Road 5', ST_LineFromText('linestring(16 0, 16 4, 16 8)'));

```

### 1.1.2 Moving to the Topology-Geometry Model

Some real-world things may have interrelationships, for example, land parcels share boundaries with other land parcel (or other things, like bodies of water) and do not overlap. [H2168R2] presents a data model to capture these relationships in a Topology-Geometry with faces, edges and nodes and Topology-Network models with netnodes and netlinks. Using this model, a user could model the same real-world things using the face, edge and node ids in a given topology-name Topology-Geometry. Here we present a possible set of tables based on the ones Subclause 1.1.1, "Modeling with the existing Geometry Model" using the face id, edge id or node id to identify the particular topology-geometry entity. Note: Since there may be different <topology-name> schema, these tables include a column to maintain which schema contains that face. If all the topology entities in the same <topology-name>, then the schema name could be maintained else where in some meta-data.

```

CREATE TABLE Park (
    PARK_NAME CHARACTER(16),
    TOPOLOGY_NAME CHARACTER(18),
    FACE_ID INTEGER
);

CREATE TABLE Parcel (
    PARCEL_ID CHARACTER(16),
    TOPOLOGY_NAME CHARACTER(18),
    FACE_ID INTEGER
);

CREATE TABLE Road (
    ROAD_NAME CHARACTER(20),
    TOPOLOGY_NAME CHARACTER(18)
    EDGE_ID INTEGER
);

CREATE TABLE Fountain (
    FOUNTAIN_NAME CHARACTER(16),
    TOPOLOGY_NAME CHARACTER(18),
    NODE_ID INTEGER
);

```

To create a face representing a Park A or Road 5 cannot be done with a single insert statement as in Subclause 1.1.1, "Modeling with the existing Geometry Model". It was our intension in the following examples is to try use the functions described in [H2168R2] to create the faces, edges and nodes need to represent Park A and Road 5. In Example (1), Park A is created in the 'Example' Topology-Geometry and in Example (2), Road 5 is then created.

Note: the examples are done as best as we could from the descriptions in the Concepts section of [H2168R2]. We needed to use some functions that have some identified outstanding issues. We identified some of the problem in notes to the reader. We also make assumptions on the ID values generated by the routines that we have identified in Notes to the reader.

### 1.1.2.1 Example (1)

Here is an example (1) where Park 'Park A' represented by 'polygon((8 8,20 8, 20 4, 8 4, 8 8))' is added to the empty Topology-Geometry schema named 'Example'.

```

DECLARE NodeId INTEGER;
DECLARE EdgeId INTEGER;
DECLARE RightFaceId INTEGER;
DECLARE LeftFaceId INTEGER;

-- We assume empty tables in the 'Example' Topology-Geometry.

-- Invoke ST_AddIsoNode the create the node represented
-- by 'point(8 8)'
SET NodeId = ST_AddIsoNode('Example', ST_PointFromText('point(8 8)'));

```

*Note to reader: we assume ST\_AddIsoNode creates node 1 (NodeId)*

```

-- Invoke ST_AddEdge to create the face representing
-- by 'polygon((8 8,20 8, 20 4, 8 4, 8 8))'

```

*Note to reader: the following SET statement is not valid SQL, without the anticipated technical proposal, its hard to know how the interface for ST\_AddEdge will be specified to return th edge id and possibly the right and left face id to the caller.*

```

SET EdgeId, LeftFaceId, RightFaceId = ST_AddEdge('Example', NodeId1,
        NodeId2, ST_LineFromText('linestring(8 8,20 8, 20 4, 8 4, 8 8)'));

```

*Note to reader: we assume ST\_AddEdge creates edge 1(EdgeId) and the ether was effectively split into the left face of 0 (LeftFaceId) and the right face of 1 (RightFaceId which represents Park A)*

```

-- Now we insert a row into the Park table for Park A
INSERT INTO Park(PARK_NAME, TOPOLOGY_NAME, FACE_ID)
        VALUES ('Park A', 'Example', RightFaceId);

```

So now the topology tables should contain:

Example.ST\_NODE

ST_NodeId	ST_FaceId	ST_Geometry (in WKT)
1	<i>the null-value</i>	'point(8 8)'

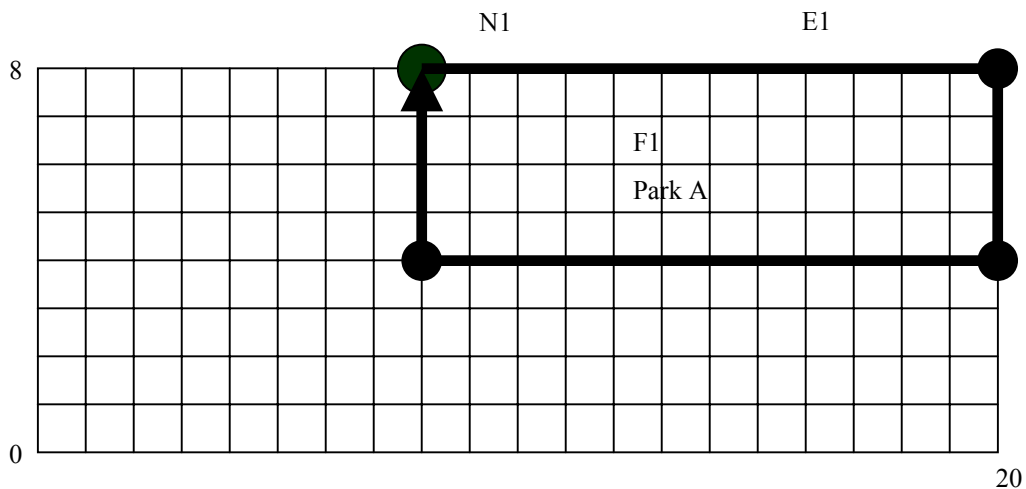
Example.ST\_EDGE

ST_Edge Id	ST_Start Node	ST_End Node	ST_Next Lface EdgeId	ST_Next Rface EdgeId	ST_Left FaceId	ST_Right FaceId	ST_Geometry (in WKT)
1	1	1	1	-1	0	1	'linestring(8 8,20 8, 20 4, 8 4, 8 8)'

Example.ST\_FACE

ST_FaceId	Geometry
0	...
1	...

Diagram with Park A



1.1.2.2 Example (2)

In this example (2), we tackle the problem of adding Road 'Road 5' represented by 'linestring(16 0, 16 4, 16 8)' to the Topology-Geometry scheme named 'Example' in the state from the last example.

```

DECLARE NodeId1 INTEGER;
DECLARE NodeId2 INTEGER;
DECLARE NodeId3 INTEGER;
DECLARE EdgeId1 INTEGER;
DECLARE EdgeId2 INTEGER;
DECLARE EdgeId3 INTEGER;
DECLARE EdgeId4 INTEGER;
DECLARE EdgeId5 INTEGER;
DECLARE RightFaceId INTEGER;
DECLARE LeftFaceId INTEGER;
    
```

```
-- Because the linestring intersects with the existing topology
-- entities in Topology-Geometry scheme named 'Example',
-- we assume the user/application has figured out that
-- an edge represented by 'linestring(16 8, 16 4)' will split
-- face 1 so the following code creates two edges represented
-- by 'linestring(16 8, 16 4)' and 'linestring(16 4, 16 0)'
-- thus splitting face 1 into two new faces.
--
-- Also, to add the edge represented by 'linestring(16 8, 16 4)',
-- edge 1 needs to be split twice, at the points represented by
-- 'point(16 8)' and 'point(16 4)'.

-- Invoke ST_SplitEdge to split edge 1 at the point represented
-- by 'point(16, 8)'.
```

*Note to reader: the following SET statement is not valid SQL, without the anticipated technical proposal, it is hard to know how the interface for ST\_SplitEdge will be specified to return a node id and the two edge ids to the caller.*

```
SET NodeId1, EdgeId1, EdgeId2 = ST_SplitEdge('Example', 1,
      ST_PointFromText('point(16 8)'));
```

*Note to reader: we assume ST\_SplitEdge creates node 2 (NodeId1), deletes edge 1 and then creates two new edges: 2 (EdgeId1 represented by 'linestring(0 0, 16 8)') and 3 (EdgeId2 represented by 'linestring(16 8, 8 8)').*

```
-- Invoke ST_SplitEdge to split EdgeId2 at the point represented
-- by 'point(16 4)'
```

*Note to reader: the following SET statement is not valid SQL, without the anticipated technical proposal, it is hard to know how the interface for ST\_SplitEdge will be specified to return a node id and the two edge ids to the caller.*

```
SET NodeId2, EdgeId2, EdgeId3 = ST_SplitEdge('Example', EdgeId2,
      ST_PointFromText('point(16 4)'));
```

*Note to reader: we assume ST\_SplitEdge creates node 3 (NodeId2), deletes edge 3 and then creates two new edges: 4 (EdgeId2 represented by 'linestring(16 8, 16 4)') and 5 (EdgeId3 represented by 'linestring(16 4, 8 8)').*

```
-- Invoke ST_AddEdge to create the edge from NodeId1 to NodeId2
```

*Note to reader: the following SET statement is not valid SQL, without the anticipated technical proposal, its hard to know how the interface for ST\_AddEdge will be specified to return the edge id and possibly the right and left face id to the caller.*

```
SET EdgeId4, RightFaceId, LeftFaceId = ST_AddEdge('Example', NodeId1,
      NodeId2, ST_LineFromText('linestring(16 8, 16 4)'))
```

*Note to reader: we assume ST\_AddEdge creates edge 6 (EdgeId4) and splits face 1, so face 1 is deleted and face 2 (RightFaceId represented by 'polygon(8 8, 16 8, 16 4, 8 4, 8 8)') and 3 (LeftFaceId represented by 'polygon(16 8, 20 8, 20 4, 16 4, 16 8)') are created.*

```
-- Invoke ST_AddIsoNode the create the node represented
-- by 'point(16 0)'
SET NodeId3= ST_AddIsoNode('Example', PointFromText('point(16 0)));
-- Note to reader: we assume ST_AddIsoNode creates node 4
```

*Note to reader: we assume ST\_AddIsoNode creates node 4 (NodeId3)*

-- Invoke ST\_AddEdge to create the edge from NodeId2 to NodeId3

*Note to reader: although the following SET statement is valid SQL, without the anticipated technical proposal, its hard to know how the interface for ST\_AddEdge will be specified to return the edge id and possibly the right and left face id to the caller.*

```
SET EdgeId5 = ST_AddEdge('Example', NodeId2, NodeId3
    ST_LineFromText('linestring(16 4, 16 0)'))
```

*Note to reader: we assume ST\_AddEdge creates edge 7 (EdgeId5) and does not split a face and that node 4 is no longer an isolated node.*

So now the topology tables should contain:

**Example.ST\_NODE**

ST_NodeId	ST_FaceId	ST_Geometry (in WKT)
1	<i>the null-value</i>	'point(8 8)'
2	<i>the null-value</i>	'point(16 8)'
3	<i>the null-value</i>	'point(16 4)'
4	<i>the null-value</i>	'point(16 0)'

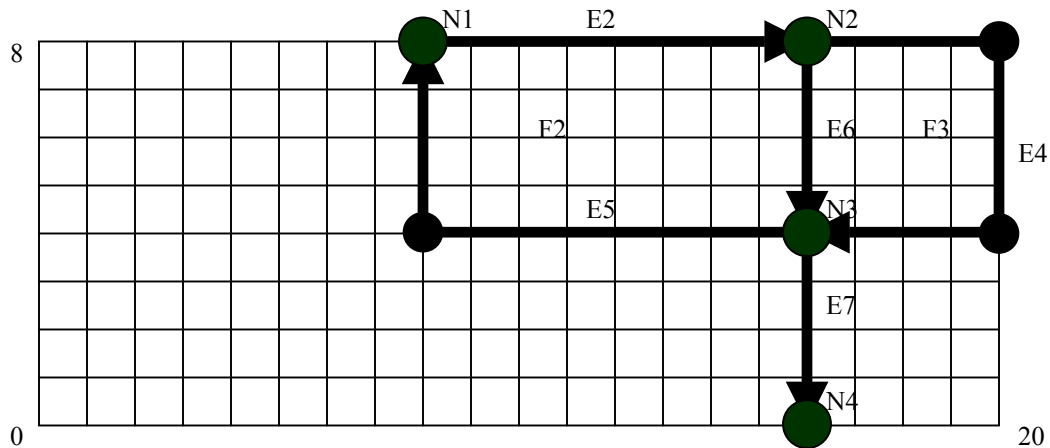
**Example.ST\_EDGE**

ST_Edge Id	ST_Start Node	ST_End Node	ST_Next Lface EdgeId	ST_Next Rface EdgeId	ST_Left FaceId	ST_Right FaceId	ST_Geometry (in WKT)
2	1	2	4	-5	0	2	'linestring(8 8,16 8)'
4	2	3	7	6	0	3	'linestring(16 8, 20 8, 20 4, 16 4)'
5	3	1	2	-6	0	2	'linestring(16 4, 8 4, 8 8)'
6	2	3	-4	-2	3	2	'linestring(16 8, 16 4)'
7	3	4	-7	5	0	0	'linestring(16 4, 16 0)'

**Example.ST\_FACE**

ST FaceId	Geometry
0	...
2	...
3	...

Diagram with faces and edges for Park A and Road 5



At this juncture, the topology representation of Park A is F2 and F3 and the topology representation of Road 5 is E6 and E7.

Recall the initial, and perhaps naive definition of the Park and Road table.

```
CREATE TABLE Park (
  PARK_NAME CHARACTER(16),
  TOPOLOGY_NAME CHARACTER(18),
  FACE_ID INTEGER
);

CREATE TABLE Road (
  ROAD_NAME CHARACTER(20),
  TOPOLOGY_NAME CHARACTER(18)
  EDGE_ID INTEGER
);
```

Previously we inserted a row in the Park table with a FACE\_ID of 1. Unfortunately, after, when we added the edges for Road 5, we split face one into face 2 and 3 (which invalidated the row for Park A in the Park table). The current definition of the Park table cannot manage the possibility that faces are inserted and deleted as more data is inserted in the Topology-Geometry model. Similarly, the Road table is not equipped to handle multiple edges.

For the Park table, the user/application can create its own many to many mapping table between Parks and faces and, for the Road table, the user/application can create its own many to many mapping table between Roads and edges. This places the burden on every application when adding or deleting an topology entities to the Topology-Geometry, to find all these mapping tables and update them accordingly. This makes any processing of topology a custom application; we do not think it is the intent of the work.

There is a large body of work in topology that provides generalized solutions for this problem. Since [H2168r2] is based on Mini-Topo, we went to the description in [19107] and found that along with support for faces, edges and nodes, there is generic support for features to support the mapping between the tables modeling the real-world things and the faces, edges and nodes. [H2168R2] needs to keep in harmony with [19107] and provide the equivalent tables and functions to manage face features, edge features and node features.

### 1.1.3 Contents of the Topology-Geometry tables in 'Example'

In our examples, we based the contents of the Topology-Geometry tables based on the examples in the discussion. The Concepts section was not complete in its description, especially w.r.t. the negative ID values.

### 1.1.4 Face 0

Face 0 is introduced in the discussion part of [H2168R2] but there is no description of the Concepts part of the proposal and proposal should be clear what this represents. As we tried to understand how these the topology mechanism be found it difficult to represent the two following use cases:

#### 1.1.4.1 Face 0 Use Case

Given empty 'Example' Topology-Geometry Tables, by executing the following:

```
DECLARE aNodeId INTEGER;

SET aNodeId = ST_AddNode('Example', ST_PointFromText('point(0 0)'))
ST_AddEdge('Example', aNodeId, aNodeId,
  ST_LineFromText('linestring(0 0, 0 1, 1 1, 1 0, 0 0)'))

SET aNodeId = ST_AddNode('Example', ST_PointFromText('point(2 2)'))
ST_AddEdge('Example', aNodeId, aNodeId,
  ST_LineFromText('linestring(2 2, 2 3, 3 3, 3 2, 2 2)'))
```

What does the Example.ST\_EDGE and Example.ST\_FACE contain?

#### 1.1.4.2 Isolated Face Use Case

There are isolated nodes and isolated edges but there appears to be no support for isolated faces, ie face within a face. Given an empty 'Example' Topology-Geometry Tables, execute the following:

```
DECLARE aNodeId INTEGER;

SET aNodeId = ST_AddNode('Example', ST_PointFromText('point(0 0)'))
ST_AddEdge('Example', aNodeId, aNodeId,
  ST_LineFromText('linestring(0 0, 0 3, 3 3, 3 0, 0 0)'))

SET aNodeId = ST_AddNode('Example', ST_PointFromText('point(1 1)'))
ST_AddEdge('Example', aNodeId, aNodeId,
  ST_LineFromText('linestring(1 1, 1 2, 2 2, 2 1, 1 1)'))
```

What does the Example.ST\_EDGE and Example.ST\_FACE contain?

## 1.2 Geometry Operations in Topology-Geometry

One of the aims of adding Topology to SQL/MM was expressed in [H2168r2], Subclause 1.4 Requirements, Paragraph 5:

As stated in ISO 19107 [2], topological complexes can be used to reduce the computational intensity of geometric calculations.

The topology data model does maintains certain topologic relationships to reduce the computational intensity, however the Topology-Geometry model does not provide functions for any of the geometry operations found in the existing standard, such as ST\_Area and ST\_Length of Topology-Geometry entities.

Since the Edges (and Nodes) have a Geometry component, to determine, for example, the length of an Edge in Topology-Geometry, the user needs only to invoke ST\_Length on the geometry column. So, in fact many of the geometry operations could be supported in Topology-Geometry for edges and nodes. However, there is no geometry component for a face, only a minimum bounding box (MBR) polygon (or Envelope as its currently known in the SQL/MM Standard). This MBR is provides for presumably some form indexing (which is not an area addressed by the SQL or SQL/MM standards). Keeping the geometry component for only Edges and Nodes and not Faces seems inconsistent. One may argue that the face geometry can be reconstructed at any time these geometry operations are performed. But this does not come without a computational cost; something that Topology functionality is trying reduce (as stated above).

Once the topology structure is build, like many applications, it tends to be relatively stable (i.e. there tends to be more display, query and analysis of the data verses update operations). Once the data is loaded, the user will pay the computational penalty to perform many routine operations such as getting the geometry for graphical display or spatial operations (such as ST\_Area, ST\_Centroid) and query operations with other geometry values (such as ST\_Intersects based on a geometry value on the faces in a <topology-name>). This is a major issue that needs addressing.

### 1.3 Equivalent Topology Queries

[H2168R2] concentrates on how the table structure and functions add and delete faces, edges and nodes in topology. As we saw in the last Subclause, the existing geometry model provides many spatial operations and it also provides some topology relationships. However there is nothing in [H2168R2] so show if and how a user/application would determine the following relationship between the topology entities:

- a) intersects
- b) disjoint
- c) contains
- d) within
- e) disjoint
- f) touches
- g) crosses

This needs to be discussed in the proposal.

### 1.4 Spatial Reference Systems

[H2168R2] does not discuss what the affect of basing a Topology-Geometry or Topology-Geometry models with geometry values in different spatial reference systems.

### 1.5 Creation of <topology-name> or <network-name> schemas

There are no routines in the Concepts of [H2168R2] to create the <topology-name> or <network-name> schemas.

### 1.6 Use of Storage

[H2168R2] uses the term "storage", but this is not defined in SQL or SQL/MM. The SQL term "site" would seem more appropriate.

### 1.7 Use of Heal

The term "split" is used to divide a face into two faces. Then it appears to use the term "heal" when a edge is removed and two faces now form one face. The term "heal" brings to mind a repair operation after some damage (in the case of self-healing system that are appearing in the marketplace). A better term for the converse of "split" would be "join". Either the term "heal" should be replaced with "join" or the "heal" be defined.

### 1.8 Unique Identifiers

In the discussion of [H2168R2], the paper specifies that how unique identifiers are derived will be implementation-defined. This should not be necessary because SQL supports identity columns for this purpose. We also found that the topology model gives special meaning to negative and positive ID values. The ID values are very important to the topology machinery of the topology tables and should have some discussion in the Concepts section.

## **1.9 Table Descriptions**

In the table descriptions, the columns are presented in a paragraph form using "mandatory" and "optional"

SQL/MM uses mandatory to describe the minimum set of features a conforming implementation needs to provide. In the context of tables, "optional" is used to describe a column in the paths\_table that may or may not be specified. This proposal introduces confusion its use of mandatory and optional when describing the values of these columns. The intent of this proposal is to define topologic entities with tables in SQL. In the "Changes from Previous Version", it clarifies that the columns are not optional, but this use of optional refers to allowing values in the column to be null or not-null and this application package defines a special meaning to null values. Since there is already adequate terminology to describe this behavior and the Concepts section should use this terminology.

[H2168R2] does use "optional" when discussing the parameters of a function. This is possible by overloading the function to have one version with the parameter and another version without.

## **2 Conclusion**

As we indicated at the beginning of the paper, INCITS H2-2004-168r2 has made a good start in responding to USA-P03-001. However we have identified several issues, unknowns and missing functionality that need to be resolved and in place before the topology paper moves forward.

**End of Paper.**