Detailed description

- « topology » (postgres) schema
  This schema and both admin tables are created when installing postgis_topology extension. This tables are used to manage topology at the highest level. Both tables are unique.

  – **“topology”**
    This table references all the postgres schema containing topology, plus some useful attributes.
    - **id** : a unique id for each topology
    - **name** : a human name for the topology, should also be unique, but it is not enforced.
    - **srid** : …
    - **precision** : this attributes control the vertex to vertex snapping.
    - **hasZ** : …

  – **“layer”**
    This table references all the column of type topogeometry in the database, and also the precise path to access this column of type topogeometry (postgres schema+ table name + column name).
    This table also has some attributes to control topogeometry inheritance.
    - **topology_id** : a link to the id of the topoglogy of the layer (topology.id)
    - **layer_id** : a unique id of a layer inside a topology. (topology_id,layer_id) should be unique.
    - **schema_name** : name of the postgres schema wich contains the layer
    - **table_name** : name of postgres table containing the layer (no schema prefix)
    - **feature_column**: name of the column which contains the topogeometries of the layer
    - **feature_type** : type of topogéometry : 1 : point, 2 : line, 3 : face, 4 : mixed
    - **level** : some control attribute to allow topogeometry inheritance.
    - **child_id** : some control attribute to allow topogeometry inheritance.

Detailed description

- « my_schema » (postgres) schema

  It is a schema (there can be many) with all the geometric level tables and the topogeometry level table, and a column with topogeometry.
  - topogemetry level
    - **“a_topogeom_column”**

      there can be as many as one wants.

      For a topology, the tuple (schema, table_name, column_name) should be unique.

      Every line of the topogeometry column contains a topogeometry object.
      - **topology_id** : link to the id of the topology schema of this topogeometry . Should be unique in the column
      - **layer_id** : link to the id of the layer of this topogeometry. Should be unique in the column
      - **id** : a unique id (inside a layer)for every line of the topogeometry column.
      - **feature_type** : topogeometry type : 1 : point, 2 : line, 3 : face, 4 : mixed
    - **“relation”**

      this is a unique table per (postgres) schema. It makes the correspondance between the topogeometries and the geometric level 0-1 topogéometrie <– > 0-N geometrical elements
      - **topogeo_id** : unique id of a topogeometry in a topogeometry column.
      - **layer_id** : Unique layer id of the topogeometry. the tuple (layer_id, id) allow to uniquely identify a topogeometry inside a schema.
      - **element_id** : id of the corresponding geometrical element.
      - **element_type** : type of geometrical element : 1 : point, 2 : line, 3 : face. The tuple (element_type,element_id) allow to identify uniquely the corresponding geometrical element. The type gives the table to look in (node, face, edge_data), the id identify the element.

Detailed description

- « my_schema » (postgres) schema
  - geometrical level
    It is not a real “geometrical” level, but it is the lowest level of postgis_topology.
    It is a traditionnal topological model with node/edge/face and topologicalrelations between it.
    There are 3 tables storing geometrical elements and relations.
    - **“face”**
      This table store faces. Each face is uniquely identified by an id. The geometry of a face is not stored and must be computed by using edges. Nevertheless ther is a bounding box for each face. It allows some computing acceleration.

– **face_id**	: the unique id for a face
– **mbr**	: Maximum Bounding Rectangle : lthe bounding box of a face.

    - **“node”**
      la table qui stocke les nœuds. Chaque nœud est identifié de façon unique. On stocke aussi sa géometrie au sens postgis, et un lien vers la face dans laquelle le nœud est si le nœud n’est pas sur une arrête (cas d’un point isolé)
      This table store nodes. each node has a unique id and a (postgis) geometry. There also is a link to face.face_id to allow dealing with isolated node (not on any edge)

– **node_id**	: a node unique id.
– **containing_face**	: when node is not on any edge, allow to find the face the nod eis in.
– **geom**	: (postgis) geometry of the node

    - **“edge_data”**
      This table store edge id and geom, plus the relation between low level geometrical element (node, edge, face).

– **edge_id**	: a unique id for each edge
– **geom**	: (postgis) geometry of the edge
– **start_node**	: link to the start nodes id of the edge
– **end_node**	: link to the end nodes id of the edge
– **next_left_edge**	: link to the next edge on the left side (relative orientation)
– **abs_next_left_edge**	: link to the next edge on the left side (absolute orientation)
– **next_right_edge**	: link to the next edge on the right side (relative orientation)
– **abs_next_right_edge**	: link to the next edge on the right side (absolute orientation)
– **left_face**	: link to the left face id of the edge
– **right_face**	: link to the right face id of the edge