International Journal of Innovation and Applied Studies
ISSN 2028-9324 Vol. 10 No. 1 Jan. 2015, pp. 336-350

http://www.ijias.issr-journals.org/

G-Brackets : un nouveau formalisme pour la représentation et l'interrogation des bases de données SIG

[G-Brackets : A New Formalism for Encoding and Querying GIS Databases]

Noureddine Chenfour and Hicham Amakdouf

ACT Research Team, Computer Science Department,
Faculty of Sciences Dhar El Mahraz,
Sidi Mohammed ben Abdellah University,
Fez, Morocco

ABSTRACT: In this paper, we outline the bridges that connect the area of GIS knowledge representation and its querying languages. Our work is in fact grounded in a research framework which we had founded to hang out with a pretty appropriate GIS query language that is adapted enough to this kind of information systems having a very particular aspects. Such aspects are unfortunately not yet fully exploited by the existing solutions.
In our study, we were led to consider the existing systems and standards for the representation of GIS information, and we quote in this case GML and ESRI. We have reached at the end of this study to the conviction that it is impossible to build a query language dealing with our basic aim if we ignore the representation of the GIS database to query.
Convinced of the robustness of the XML representation, but aware of his extremely verbose and difficult to read character, we had then to take inspiration from XML technology while correcting this verbosity problem and the underlying complexity.
We present then in this paper a new formalism that is able to interest, engage and influence the business of geographic information. Our formalism "G-Brackets" has to respond strongly to our initial aims which can be summarized in the following goals:
- A solid and hierarchical formalism with the Manner of XML technology.
- An object oriented representation.
- A knowledge based representation according to first-order predicates logic, ready to be requested by an inference engine able to go beyond an ordinary SQL engine.
- In the end, a simple and less verbose symbolism that is readable and able to separate between the data model and its semantics. Such pattern has constituted for us a real extension of the MVC design pattern that we named MVX pattern.

KEYWORDS: GIS, GML, XML, Spatial Databases, Query Language.

RESUME: Dans cet article, nous esquissons les ponts qui rapprochent le domaine de la représentation de la connaissance SIG et celui de son interrogation. Le travail se situe en fait dans un cadre de recherche que nous avions tracé pour sortir avec un langage d'interrogation SIG assez approprié pour ce genre de systèmes d'information ayant une spécificité très particulière. Une telle spécificité n'est malheureusement pas encore pleinement exploitée par les solutions existantes.
Au cours de notre étude, nous avions été amené à se pencher sur les systèmes et les standards existants permettant la représentation des informations SIG, et nous citons en l'occurrence GML et ESRI. Nous avons abouti au terme de cette étude

Corresponding Author: Noureddine Chenfour

à la conviction qu'il est impossible de sortir avec un langage d'interrogation approchant notre objectif de base si nous faisons abstraction de la représentation de la base de données SIG à interroger.
Convaincu de la robustesse de la représentation XML, mais conscient de son caractère extrêmement verbeux et difficilement lisible, nous devions alors nous inspirer de la technologie XML tout en corrigeant ce problème de verbosité et la complexité sous-jacente.
Nous présentons alors dans cet article un formalisme susceptible d'intéresser, voire d'impliquer et influencer les métiers de l'information géographique. Notre formalisme « G-Brackets » a permis de répondre très fortement à nos objectifs initiaux que nous pouvons résumer dans les points suivants :
- Un formalisme solide et hiérarchique à la manière de la technologie XML.
- Une représentation orientée objet.
- Une représentation basée sur les connaissances et la logique des prédicats de premier ordre, s'apprêtant à être interrogée par un moteur d'inférence capable d'aller au-delà d'un moteur SQL ordinaire].
- En fin un symbolisme simple, moins verbeux, lisible et séparant modèle et sémantique des données ; ce qui constituera pour nous une véritable extension du design pattern MVC et que nous avons nommé MVX.

MOTS-CLEFS: SIG, GML, XML, bases de données spatiales, langage de requête.

1 INTRODUCTION

La technologie XML, qui constitue une véritable galaxie et un croisement de grandes technologies, a toujours eu comme principal inconvénient la verbosité. Un tel inconvénient est généralement négligé devant le grand apport de la technologie XML, sa flexibilité, son caractère hiérarchique et bien d'autres grands avantages. Depuis la fin des années 90, toutes les disciplines et technologies existantes ont commencé à intégrer le « formalisme » XML pour le stockage des données, le paramétrage des applications et des plateformes ainsi que l'échange de données.

Nous avons aujourd'hui de grandes solutions complètement basées sur XML [1] avec une grande satisfaction.

Cependant, lorsqu'on explore le domaine des systèmes d'information géographiques (SIG), et principalement les standards de l'OGC (Open Geospatial Consortium) et en l'occurrence le langage GML (Geography Markup Language) servant à la représentation des données SIG [2], [3], [4] la verbosité est tellement éminente qu'elle influence très fortement sur la lisibilité du contenu et rend rapidement la technologie pas très adaptée à ce genre de systèmes.

Analysons ainsi le fragment GML suivant décrivant une entité géographique, tirée d'une zone industrielle de la ville de Fès, par l'intermédiaire de son identificateur (9), son nom « *Technologie et Services pour Tannerie* » ainsi que sa géométrie à l'aide des cordonnées (x, y) du polygone qui la délimite :

```
<gml:featureMember>
      <ms:polygon fid="">
            <gml:boundedBy>
                  <gml:Box srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
                        <gml:coordinates>
                              28.929774956042,9.46613324338279
                              39.403875227383914,17.399335343386387
                        </gml:coordinates>
                  </gml:Box>
            </gml:boundedBy>
            <ms:msGeometry>
                  <gml:Polygon name="Technologie et Services pour Tannerie"
                        srsName="http://www.opengis.net/def/crs/EPSG/0/4326">
```

```
                        <gml:outerBoundaryIs>
                              <gml:LinearRing>
                                    <gml:coordinates>
                                          39.403875227383914,13.192757810419211
                                          34.435042470010345,9.46613324338279
                                          32.66904390493049,10.860820931531418
                                          28.929774956042,13.813861905587784
                                          33.75745658351524,17.399335343386387
                                          37.56563829800576,14.56224565672999
                                          39.403875227383914,13.192757810419211
                                    </gml:coordinates>
                              </gml:LinearRing>
                        </gml:outerBoundaryIs>
                  </gml:Polygon>
            </ms:msGeometry>
            <ms:ogc_fid />
            <ms:name>Technologie et Services pour Tannerie</ms:name>
            <ms:id>9</ms:id>
      </ms:polygon>
</gml:featureMember>
```

Il s'agit d'une représentation très bien détaillée mais trop complexe, et la méta information est plus volumineuse que l'information en elle-même comme le démontre le tableau ci-dessous :

Tableau 1. Pourcentage de l'information de description GML

Nombre total de caractères (espaces non compris)	868
Nombre de caractères composant l'information véhiculée par ce flux XML	277
Pourcentage d'information de description (*superflue*)	68,09 %

Le tableau 1 montre que seule 32% du flux constitue l'information SIG utile ; tout le reste est de l'information de description du contenu.

Si nous analysons la représentation XML d'ESRI (Environmental Systems Research Institute), nous pouvons tirer les mêmes conclusions que celles constatées avec GML, et nous aboutissons donc aux mêmes inconvénients, avec un taux de verbosité plus avéré.

Nous présentons alors le même cas d'étude avec en plus quelques données attributaires : Secteur d'activité, Gérant, Numéro de la zone et Observation.

Ci-dessous la représentation ESRI de l'entité :

```
<Record xsi:type="esri:Record">
      <Values xsi:type="esri:ArrayOfValue">
            <Value xsi:type="xs:int">9</Value>
            <Value xsi:type="esri:PolygonN">
                  <HasID>false</HasID>
                  <HasZ>false</HasZ>
                  <HasM>false</HasM>
                  <Extent xsi:type="esri:EnvelopeN">
                        <XMin>533696.25029999763</XMin>
```

```
                        <YMin>376744.76209999993</YMin>
                        <XMax>533755.1466999976</XMax>
                        <YMax>376789.37089999992</YMax>
                    </Extent>
                    <RingArray xsi:type="esri:ArrayOfRing">
                        <Ring xsi:type="esri:Ring">
                            <PointArray xsi:type="esri:ArrayOfPoint">
                                <Point xsi:type="esri:PointN">
                                    <X>533755.14670000225</X>
                                    <Y>376765.71709999815</Y>
                                </Point>
                                <Point xsi:type="esri:PointN">
                                    <X>533727.20669999719</X>
                                    <Y>376744.76209999993</Y>
                                </Point>
                                <Point xsi:type="esri:PointN">
                                    <X>533717.27639999986</X>
                                    <Y>376752.60449999943</Y>
                                </Point>
                                <Point xsi:type="esri:PointN">
                                    <X>533696.25029999763</X>
                                    <Y>376769.20960000157</Y>
                                </Point>
                                <Point xsi:type="esri:PointN">
                                    <X>533723.39660000056</X>
                                    <Y>376789.37090000138</Y>
                                </Point>
                                <Point xsi:type="esri:PointN">
                                    <X>533744.81019999832</X>
                                    <Y>376773.4178000018</Y>
                                </Point>
                                <Point xsi:type="esri:PointN">
                                    <X>533755.14670000225</X>
                                    <Y>376765.71709999815</Y>
                                </Point>
                            </PointArray>
                        </Ring>
                    </RingArray>
                </Value>
                <Value xsi:type="xs:string">
                    Technologie et Services pour Tannerie
                </Value>
                <Value xsi:nil="true" />
                <Value xsi:type="xs:string">services pour les tanneurs</Value>
                <Value xsi:type="xs:string">Thami Amraoui</Value>
                <Value xsi:type="xs:string">Travaux En Cours</Value>
                <Value xsi:nil="true" />
                <Value xsi:nil="true" />
                <Value xsi:type="xs:short">101</Value>
                <Value xsi:type="xs:double">147.77765548177521</Value>
                <Value xsi:type="xs:double">1306.9949917962801</Value>
            </Values>
</Record>
```

Tableau 2. Pourcentage de l'information de description XML d'ESRI

Nombre total de caractères (espaces non compris)	1552
Nombre de caractères composant l'information véhiculée par ce flux XML	372
Pourcentage d'information de description (*superflue*)	76,03 %

Donc par rapport à ce bilan, seul 24% du flux constitue l'information qu'on souhaite véhiculer. Les 76% sont utilisés pour envelopper et décrire l'information.

On pourra rajouter à ça que la représentation XML en elle-même n'est généralement utilisée dans les SIG que pour jouer le rôle de médiateur de données et reste donc utilisée principalement pour échanger des données entre deux systèmes hétérogènes. La plupart des solutions SIG existantes utilisent des bases de données relationnelles pour la représentation des données attributaires [5]. Cette réalité a toujours influencé et guidé le choix d'un langage d'interrogation SIG [6], [7]. Dans ce cas encore, la plupart des solutions existantes se basent exclusivement sur le langage SQL [8], [9], [10], [11] *qui n'a jamais été créé pour interroger des bases de données SIG!* Il ne dispose alors d'aucune représentation optimisée relativement à la nature du contenu traité.

Nous nous trouvons donc dans une situation qui nécessite aussi bien une réflexion sur la représentation que sur l'interrogation d'un contenu SIG [12].

Nous proposons alors dans cet article un formalisme consubstantiel qui nous permettra la représentation, le stockage ainsi que l'interrogation de contenu SIG. Il s'agit d'un formalisme inspiré de 3 Concepts :

- JSON comme concurrent standard de la solution XML et permettant principalement d'atténuer le problème de la verbosité.
- Formalisme Objet permettant de représenter les entités, les couches et les différentes composantes SIG en les considérons des objets ce qui devrais être naturellement le cas (Every Thing is an Object).
- Logique des prédicats de premier ordre en intelligence artificielle nous permettant ainsi de considérer un contenu SIG comme une base de connaissances acceptant en conséquence d'être interrogée à travers un moteur d'inférence [13] ; ce qui permettrai à un SIG de déduire plus de connaissances SIG que ce que déduirais une solution classique basée sur SQL [14].

2 FORMALISME PROPOSÉ

L'objectif est de définir un formalisme objet permettant à la fois de représenter les composantes manipulées dans un SIG mais aussi de modéliser un système d'interrogation approprié.

Avant de définir la syntaxe générale du formalisme SIG « G-*Brackets* » proposé dans cet article et basé lui-même sur un formalisme plus général que nous avons nommé « Brackets », nous considérons un sous ensemble de composantes SIG à base desquelles nous allons faire un montage symbolique que nous généralisons par la suite.

Les systèmes d'information géographique sont principalement basés sur les 4 composantes, que nous allons nommer « Classes Géographiques de Base » (Basic Geographical Classes) et pouvant aussi être vues comme des prédicats : Entité géographique (Geographical Entity), Couche (Layer), Carte (Map) et Système d'information géographique (GIS).

2.1 SYMBOLISME D'UNE ENTITÉ GÉOGRAPHIQUE

2.1.1 REPRÉSENTATION

Représentée par l'intermédiaire de la classe géographique « ***Entity*** », une entité est définie comme suit :

Entity(id, name, address, x1, y1, x2, y2, …, Layer(id), Attributes(a1, a2, …))

- ⇨ Avec **id, name, address, x1, y1, x2, y2,** … sont les propriétés géo spatiales constituant la partie fixe et commune à tous les SIG.
- ⇨ Tandis que **a1, a2,** … constituent la partie optionnelle représentent les propriétés attributaires qui changent avec le SIG à modéliser.

Exemple :

Entity(9, Technologie et Services pour Tannerie, ,
533755.14, 376765.71, 533727.20, 376744.76, 533717.27, 376752.60, 533696.25,
376769.20, 533723.39, 376789.37, 533744.81, 376773.41, 533755.14, 376765.71,
Attributes (
Technologie et Services pour Tannerie,
services pour les tanneurs,
Thami Amraoui,
101,
Travaux En Cours
)
)

Les données attributaires sont par exemple dans ce cas respectivement : « Company Name », « Activity Sector », « Manager », « Area Number », « Observation ».

Il est aussi à remarquer que comparativement à la notation JSON (JavaScript Object Notation), nous gagnons aussi en verbosité, en redondance et en séparation entre la sémantique et le modèle de données. Nous rappelons à cet effet que JSON [15] utilise la notation « "Name" : "Value" ». Tandis que dans notre formalisme « Brackets » les noms des propriétés, qui portent avec eux la sémantique des données, ne sont aucunement entrelacés avec les valeurs. Seules ces dernières sont représentées dans la base de connaissances (les guillemets sont aussi éliminées dans « Brackets »).

Ci-dessous un exemple de représentation de données en JSON :

```
{
    "entities" : [
      {"id" : "101", "name" : "Brother Style", "manager" : "El Hhalifi",  "area" : "1311.09" },
      {"id" : "102", "name" : "Grains Bouyeblane", "manager" : "Boujemaa", "area" : "708.32" },
      {"id" : "103", "name" : " ROCSIGN", "manager" : "Ben Makhlouf", "area" : "806.72" },
     ]
}
```

Nous constatons que les noms des propriétés sont repris dans toutes les occurrences d'objets de même type. Le calcul du nombre de caractères de ce flux de données, sans compter les espaces, donne une valeur de 235 caractères.

Si on voudrait reprendre ces mêmes données avec une représentation XML, nous pouvons proposer la structure suivante :

```
<entities>
    <entity>
        <id>101</id>
        <name>Brother Style</name>
        <manager>El Hhalifi</manager>
        <area>1311.09</area>
    </entity>
    <entity>
        <id>102</id>
        <name>Grains Bouyeblane</name>
        <manager>Boujemaa</manager>
        <area>708.32</area>
    </entity>
    <entity>
        <id>103</id>
        <name>ROCSIGN</name>
        <manager>Ben Makhlouf</manager>
        <area>806.72</area>
    </entity>
</entities>
```

La taille de ce flux est de 325 caractères. On remarquera que JSON résout juste le problème de la balise fermante, mais il garde le problème de la redondance.

Une représentation XML plus optimale pourrait être définie comme suit :

```
<entities>
        <entity id = "101" name = "Brother Style" manager = "El Hhalifi" area = "1311.09" />
        <entity id = "102" name = "Grains Bouyeblane" manager = "Boujemaa" area = "708.32" />
        <entity id = "103" name = "ROCSIGN" manager = "Ben Makhlouf" area = "806.72" />
</entities>
```

Avec cette nouvelle représentation basée sur les attributs nous avons pu ramener la taille du flux à 226 caractères, qui est dans ce cas une valeur inférieure à celle obtenu avec une représentation JSON !

Si nous voudrions représenter les mêmes données avec notre formalisme *Brackets* ce sera défini comme suit :

```
Entity(101, Brother Style, El Hhalifi, 1311.09)
Entity(102, Grains Bouyeblane, Boujemaa, 708.32)
Entity(103, ROCSIGN, Ben Makhlouf, 806.72)
```

Avec une taille de 124 caractères et donc une réduction de plus de 47% par rapport à son équivalent JSON.

Il est cependant évident que le système Brackets nécessite une partie méta données dans laquelle on devrait fournir une définition des noms et des types associés aux données attributaires [16]. Cette contrainte reste cependant naturelle et compatible avec les DTDs de la technologie XML.

Néanmoins, Il n'est nullement nécessaire de démontrer la lisibilité (ou la clarté) et la concision attachées à cette représentation. Le tableau ci-dessous confirme clairement à quel point cette représentation est concise sur l'exemple SIG initial.

Tableau 3. Pourcentage de l'information de description G-Brackets

Nombre total de caractères (espaces non compris)	286
Nombre de caractères composant l'information véhiculée par ce flux	245
Pourcentage d'information de description ($superflue$)	14,34 %

Nous relevons un taux extrêmement optimisé de ***14,34 %*** d'information de description contre ***76 %*** avec XML d'ESRI et ***68 %*** avec GML. Il est aussi à noter que pour véhiculer la même information G-Brackets utilise un flux de 286 caractères contre 1552 avec XMl d'ESRI ; ce qui ne représente que ***18,43 %*** du flux ESRI et donc un gain de ***81,57 %.***

Avec cette représentation, on sépare entre la structure ou sémantique du contenu et le modèle ou le flux de données. Ces deux aspects sont habituellement entrelacés dans un formalisme XML ou JSON.

Il reste cependant à prouver que ce formalisme offre une couverture totale relativement à toutes les représentations XML existantes ; ce que nous confirmons être le cas d'après toutes les études de cas et les expériences que nous avons menées sur ce domaine.

2.1.2 INTERROGATION

L'interrogation pourra alors être réalisée tout naturellement sur la base de la même représentation.

Nous définissons un ensemble de règles basées sur le même formalisme tel que décrit précédemment.

Ci-dessous des exemples :

Tableau 4. Exemples de requêtes spatiales G-Brackets

Entity(9)	Récupérer un objet de type « Entity » correspondant à l'entité dont l'identificateur est égal à 9.
Entity(, , , 533755.14, 376765.71, ..)	Récupérer la liste des entités dont le premier point est (533755.14, 376765.71).
Entity(.. , 533755.14, 376765.71, ..)	Récupérer la liste des entités contenant le point (533755.14, 376765.71)
Entity(Attributes(..,Thami Amraoui, ..))	Récupérer la liste des entités dont l'une des données attributaires a pour valeur « Thami Amraoui »
Entity(Attributes(, ,*Thami*, ..))	Récupérer la liste des entités dont le 3ème attribut contient la valeur « Thami »
Entity(Attributes(!Thami, ..))	Récupérer la liste des entités dont la valeur du premier attribut est différente de la chaîne « Thami »
Entity(Attributes(..,!Thami, ..))	Récupérer la liste des entités dont aucun attribut n'est égal à la chaîne « Thami »
Entity(Layer(20))	Récupérer la liste des entités dont appartenant à la couche d'identificateur 20

Concernant la recherche d'entités géographiques avec des requêtes basées sur des contraintes d'extraction particulières, nous utilisons alors le même formalisme avec une syntaxe qui ressemble aux assertions suivantes :

Entity(Contains(…))

Entity(Inside(…))

Entity(Intersect(…))

En règle générale, pour extraire des entités à base de contraintes on utilisera la syntaxe suivante :

Entity(C1(…), C2(…), …)

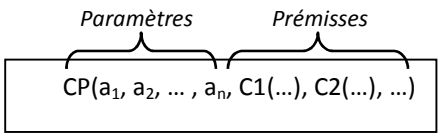
Telle que C1, C2, … sont des contraintes (ou des prédicats) à satisfaire par les entités recherchées. Le modèle est en plus insensible à l'ordre d'apparition des contraintes. Il est aussi à remarquer que « Attributes » et « Layer » étaient précocement des contraintes. Il s'agit de contraintes par défaut figurant basiquement dans la syntaxe de définition des entités. Ainsi, l'écriture suivante issue du tableau 4 :

Entity(Attributes(, ,*Thami*, ..))

peut être lue : Extraire la liste des entités telles que la contrainte « Attributes » est satisfaites. Cette dernière pose une contrainte sur son troisième argument. A ce propos, chaque contrainte a une arité propre, définie en fonction de la sémantique véhiculée par celle-ci.

Nous définissons une liste de contraintes intrinsèques G-Brackets permettant de répondre aux exigences interrogatives habituelles : Contains, Inside, Intersect, etc.

La syntaxe G-Brackets offre aussi la possibilité de créer des contraintes personnalisées, définies comme déductions logiques d'une combinaison de contraintes intrinsèques :



Cette syntaxe permet de définir une contrainte personnalisée d'arité n : $CP(a_1, \ldots, a_n)$, qui est considérée satisfaite si les prémisses C1(…), C2(…), … sont satisfaites.

Le modèle de définition des contraintes est structuré selon le design pattern « Composite » :

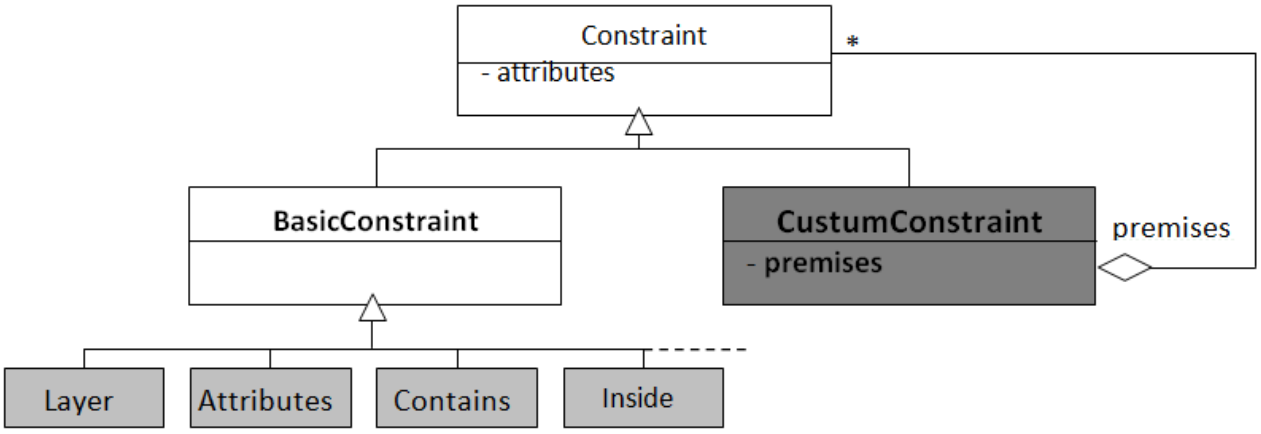


Fig. 1 : Modèle de composition des contraintes personnalisées

2.1.3 EVALUATION

Afin d'évaluer notre formalisme G-Brackets pour le cas de l'interrogation, nous nous proposons d'analyser les échantillons de requêtes spatiales suivantes [17], [18] :

1. Chercher la liste des entités contenant le polygone ((10, 12), (10, 15), (15, 25), (15, 15), (10, 12))

Représentation PostGis [19] :

SELECT id, geometry FROM table

WHERE

geomety && 'POLYGON((10 12, 10 15, 15 25, 15 15, 10 12))'

AND

Contains(geometry, 'POLYGON((10 12, 10 15, 15 25, 15 15, 10 12))';

➔ 136 caractères

Formalisme G-Brackets :

Entity(Layer(table), Contains(10, 12, 10, 15, 15, 25, 15, 15, 10, 12**))**

➔ 60 caractères

2. Chercher tous les objets dans un rayon de 250 mètres autour du point (2000, 3000)

Représentation PostGis :

SELECT * FROM table

WHERE ST_DWithin(colonne, 'POINT(2000 3000)', 250);

➔ 63 caractères

La fonction ST_DWithin(geometry, geometry, distance) permet de réaliser un calcul de distance utilisant les index.

Formalisme G-Brackets :

Entity(layer(table), Inside(Circle(2000, 3000, 250)))

➔ 50 caractères

Avec Circle un objet géométrique parmi une liste d'objets géométriques définis par G-Brackets.

3. Chercher tous les objets de la table « A » en intersection avec l'entité « X » de la table B

Avec MAPInfo

Select * from A, B where A.Obj intersects B.Obj and B.TOPONYMIE = "X"

Formalisme G-Brackets :

Entity(Layer(A), Intersects(Entity(X, Layer(B))))

2.1.4 ENTITÉS SYNTHÉTIQUES

La modélisation G-Bracket offre un certain nombre d'« opérations géographiques » permettant de trouver ou extraire des points particuliers depuis une ou plusieurs couches. Nous prenons à titre d'exemple le calcul d'un chemin reliant deux entités géographiques. Le résultat de cet exemple de requêtes sera constitué d'un ensemble de points de repère par lesquels le chemin passe. Ce résultat sera ainsi représenté comme une entité linéaire logique générée pour représenter le chemin. Nous appelons ce genre d'entité « **Entité Synthétique** ». L'opération géographique qui permet de calculer le chemin entre deux entités est définie à l'aide la requête suivante :

Path (Entity(id1), Entity(id2))

Il est aussi à remarquer que les entités synthétiques peuvent subir les mêmes opérations que peut accepter toutes autres entités ordinaires ; exemple :

Distance(Path(Entity(id1)), Entity(id2)), Path(Entity(id3)), Entity(id4)))

2.1.5 SYNTHÈSE

La gestion des entités géographiques, en tant que classes fondamentales parmi les « ***Classes Géographiques*** », a dégagé d'autres dépendances en terme de modélisation G-Brackets : Les « ***Contraintes*** », les « ***Objets Géométriques*** » et les « ***Opérations Géométriques*** ». Ce paquet d'objets a ainsi été étudié selon les deux facettes : Modélisation et interrogation, tout en démontrant de grands avantages en lisibilité, en concision et en simplicité de codage sans oublier la couverture des différents besoins SIG habituellement assurés par les grands systèmes existants.

2.2 SYMBOLISME D'UNE COUCHE

2.2.1 REPRÉSENTATION

Représentée par l'intermédiaire de la classe « ***Layer*** », une couche est définie comme suit :

Layer (id, name, DTD(…), E1Id, E2Id, …)

Avec :

- id : identificateur de la couche
- name : nom de la couche
- DTD(…) : la définition des types de données (métadonnées) utilisés par la couche. Il existe 3 façons de définir une DTD : Locale, Interne et Externe. Nous allons donner plus de détails sur les DTD dans le paragraphe 3.
- E1Id, E2ID, … sont les identificateurs des entités qu'on aimerait faire appartenir à la couche.

On remarquera encore que la lisibilité et la concision sont bien conservées comme avantages majeurs associés au formalisme G-Brackets. Nous retenons en plus l'autonomie de la couche et sa séparation vis-à-vis des entités qui sont indiquées par référencement. Nous pouvons à ce titre conclure qu'il nous sera possible, à titre d'exemple, de faire appartenir une même entité à une ou plusieurs couches !

Il est néanmoins à noter que nous définissons aussi la notion d'« ***Entités Internes*** » que nous pouvons définir à l'intérieur de la couche elle même :

Layer (id, name, DTD(…), E1Id, ***Entity(…)***, …)

Avec, bien évidemment, l'éventualité de combiner entre les deux représentations. Nous préférons cependant le référencement des entités qui offre plus d'autonomie, de concision et de lisibilité aussi bien à la couche qu'à l'entité ; avec un caractère relationnel dynamique pouvant être détaché ou rattaché au besoin.

2.2.2 INTERROGATION

Pour interroger notre base de données G-Brackets, nous utilisons le même formalisme. Ci-dessous quelques exemples de requêtes :

Tableau 5. Exemples de requêtes spatiales G-Brackets sur les couches

Layer(* ,9, *)	Récupérer la liste des couches contenant l'entité dont l'identificateur est égal à 9.
Layer(*,Entity(,Tannerie,), *)	Récupérer la liste des couches contenant l'entité dont le nom est égal à « Tannerie ».
Layer(*,Entity(Attributes(*,Tannerie , *), *)	Récupérer la liste des couches contenant des entités dont l'une des informations attributaires quelconque est égale à « Tannerie ».

2.3 SYMBOLISME D'UNE CARTE

2.3.1 REPRÉSENTATION

Une carte, définie comme un ensemble de couches, est représentée par l'intermédiaire de la classe « ***Map*** » est pourra être formalisée comme suit :

Map (id, name, Style(…), L1Id, L2Id, …)

Nous retenons alors les mêmes avantages et caractéristiques de la définition des couches. Nous parlerons dans ce cas aussi de couches « **référencées** » ou encore au besoin les « **couches internes** ».

Map (id, name, Style(…), L1Id, Layer(…), …)

Cependant, nous avons intégré en ce niveau de carte la notion de ***style***. Les styles sont en cours de développement et ne font donc pas objet de cet article, nous les présentons dans un travail futur.

2.3.2 INTERROGATION

Les mêmes règles d'interrogation sont conservées en ce niveau se basant sur le même formalisme utilisé pour la représentation des données.

2.4 SYMBOLISME D'UN SIG

2.4.1 REPRÉSENTATION

Il s'agit du niveau le plus haut dans la hiérarchie agrégative des classes géographiques. Nous définissons ainsi un SIG comme un ensemble de cartes associées à l'application SIG entière et représenté par l'intermédiaire de la classe « ***GIS*** ». Un SIG peut être défini à l'aide du formalisme suivant :

GIS (id, name, Style(…), M1Id, M2Id, …)

Permettant de conserver les mêmes caractéristiques et particularité ; nous pensons notamment au cartes « référencées » et « cartes internes ». Dans ce cas encore, nous définissons les « cartes internes » comme suit :

GIS (id, name, Style(…), M1Id, Map(…), …)

Nous sommes en train d'étudier la possibilité d'associer des styles à toutes les classes ou objets géographiques : Entité, Couche, Carte et SIG. Ce travail, comme mentionné précédemment, est en cours d'étude.

2.4.2 INTERROGATION

Il est désormais classique de dire que les mêmes règles d'interrogation utilisées avec les autres classes géographiques sont conservées en ce niveau aussi, se basant bien évidemment sur le même formalisme utilisé pour le symbolisme de la base de données.

3 DESCRIPTION DES TYPES DE DONNEES ET MODELE MVX

Nous arrivons en fin au dernier problème à résoudre ; il s'agit de la définition des noms et de la sémantique des données permettant de caractériser les entités géographiques [20]. Le formalisme utilisé a été créé de manière à séparer complètement le modèle de données porté par une structure particulière de parenthèses vis-à-vis de la définition ou la sémantique attachée à chaque nœud ou objet contenu dans cette hiérarchie de parenthèses. Nous considérons ce modèle de représentation comme une extension du modèle MVC largement utilisé en ingénierie du logiciel et ayant prouvé sa robustesse et sa grande flexibilité. Nous avons ainsi ajouté à cette flexibilité une autre dimension : la sémantique et le nom de la donnée pour séparer entre le modèle et sa sémantique, ce qui nous permettrait à titre d'exemple de changer le nom ou la sémantique d'une donnée sans avoir à retoucher le modèle lui-même. Mais, dans notre formalisme G-Brackets, nous avons abouti à une autre conséquence d'utilisation de ce modèle, la concision dans la représentation sans aucune redondance ou reprise des noms des attributs ou paramètres.

Notre modèle ainsi baptisé « MVX » : Model-View-Controller-Semantic, sera représenté comme le montre la figure suivante :

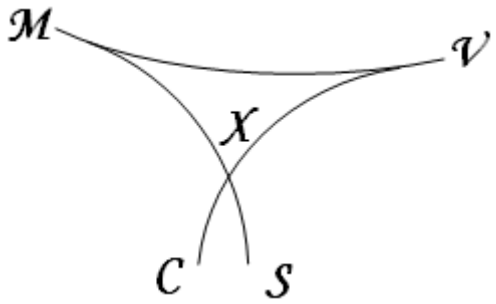


Fig. 2 : Modèle MVX utilisé par G-Brackets

Le concept de DTD sera subséquemment emprunté pour la séparation de la définition des noms et de la sémantique des données SIG utilisées par les entités géographiques. Une DTD sera définie comme suit :

DTD (id, nom,
Attributes(
a1([index,] nom, type, taille, description, statut, composition),
a2(…),
…
)
)

Exemple :

DTD (1, Zone Industrielle,
Attributes (
CompanyName(0, Company Name, String, 50, Nom de l'entreprise),
ActivitySector(1, Activity Sector, String, 30, Secteur d'activité),
Manager(2, Manager, String, 40, Gérant),
AreaNumber(3, Area Number, Integer, , Numéro de la Zone, Unique),
Observation(4, Observation, String)
)
)

La DTD permettra aussi de définir les contraintes personnalisées telles que présentées dans le paragraphe 2.1.2 :

DTD (...
CP(a_1, a_2, … , a_n, C1(…), C2(…), …),
...
)

Malgré le fait que la structure des classes géographiques est préalablement définie dans G-Brackets Core, il est aussi possible de les redéfinir à l'aide de DTD :

```
DTD (   …
        Entity (
                Id(0, Id, Integer, 10, Entity Identifier, Unique),
                Name(1, Name, String, 256, Entity Name),
                Address(2, Adress, String, 256, Entity Adress),
                Geometry(3, Geometry, List, , , ,
                        X(X, Double),
                        Y(X, Double)
                ),
                Constraints(4, Constraint, List, , , ,
                        Constraint (
                                Constraint,
                                Union, , , ,
                                Layer, Attributes, Contains, Inside, Intersect, ...
                        )
                )
        ),
        ...
)
```

Il est à ajouter que la définition des DTDs pourra être réalisée dans 3 niveaux différents :

1. DTD externe définie dans un fichier séparé et référencée depuis le fichier de définition des couches à l'aide de la syntaxe suivante :

   Layer (id, name, DTD(Id, FilePath),…)

2. DTD interne définie dans le même fichier de définition des couches. Celle-ci sera référencée à l'aide de la syntaxe suivante :

   Layer (id, name, DTD(Id),…)

3. DTD locale définie comme attribut de la couche à l'aide de la syntaxe suivante :

```
***Layer*** (id, name,
        DTD(Id, nom,
            Attributes(
                a1([index,] nom, type, taille, description, statut, composition),
                a2(…),
                …
            ),
            …
        ),
        …
)
```

4 ARCHITECTURE GENERALE DU SYSTEME G-BRACKETS

Ci-dessous une vue globale du système G-Brackets :

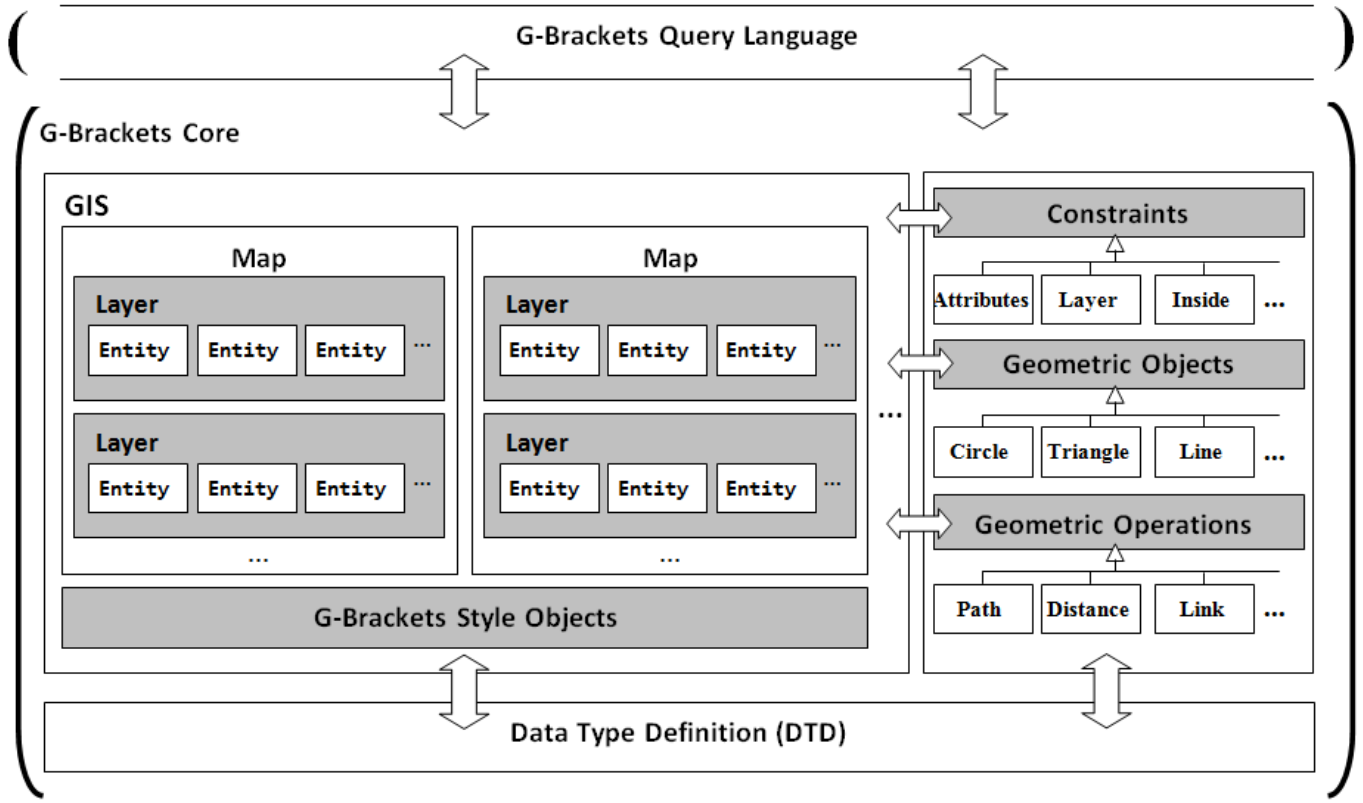


Fig. 3 : Architecture générale du système G-Brackets

Le noyaux G-Brackets est constitué principalement d'un ensemble de classes géographiques dont la définition est basée d'une part sur 3 objets G-brackets que sont les contraintes, les objets et les opérations géométriques ; d'autres part sur les DTDs définissant les noms, les types et la sémantique du contenu; sans négliger les objets de styles associés à la visualisation des cartes qui composent le système d'information géographique.

5 CONCLUSION

Dans cet article, nous avons décrit un nouveau système G-Brackets permettant aussi bien la représentation d'une base de connaissances SIG que l'interrogation du système sous jacent dans un seul et unique formalisme. Tout au long de cette étude, nous avons démontré la concision, la lisibilité, l'autonomie et la flexibilité de notre formalisme G-Brackets. Nous avons aussi démontré la couverture des principaux aspects assurés par les standards existants et en l'occurrence GML et ESRI.

A travers le Design Pattern MVX que nous avons créé par la même occasion, nous avons pu d'une part séparer la sémantique des données de leur structure de représentation et d'autre part réduire considérablement la taille de l'information de description habituellement entrelacée avec les données et présentant ainsi une grande redondance dans le langage GML et la représentation XML d'ESRI.

Notre approche nous a ainsi conduit à un modèle de représentation extrêmement solide en gardant une grande simplicité de représentation et d'interrogation SIG.

En terme d'interopérabilité, nous sommes en train de travailler sur l'interrogation de bases de données GML et ESRI à l'aide de notre formalisme G-Brackets. L'autre aspect réciproque : interroger une base de données G-Brackets avec SQL et XQuery, constitue aussi une autre ouverture d'étude de la plateforme G-Brackets.

RÉFÉRENCES

[1] World Wide Web Consortium: *XML, Extensible Markup Language*. http://www.w3.org/TR/REC-xml.

[2] Bardet, J. and Zand, A., 2009. *Spatial Modeling of Geotechnical Information Using GML*. Transactions in GIS, 13(1), 125-165. Online publication. 1-Feb-2009.

[3] Simon Cox, Adrian Cuthbert, Ron Lake, Richard Martell. Eds, *Geography Markup Language (GML) 2.0*, Open GIS Consortium Inc, 2001

[4] Gerasika, A (2011). *How to convert JSON to XML using ANTLR.* http://www.gerixsoft.com/blog/xslt/json2xml2

[5] E. Bocher, T. Leduc, G. Moreau, and F. González Cortés. GDMS: *an abstraction layer to enhance Spatial Data Infrastructures usability*. In 11th AGILE International Conference on Geographic Information Science - AGILE'2008, Girona, Spain, may 2008.

[6] Ranga Raju Vatsavai (2002) : *GML-QL, A spatial query language specification for GML*, UCGIS Summer 2002, Athens, Georgia, 2002

[7] Y. Liu, Y. Wang, Y. Zhang, X. Lin, and S. Qin. GSQL-R: *A query language supporting raster data*. In EEE International Geoscience and Remote Sensing Symposium, IGARSS, volume 7, pages 4414–4417, september 2004.

[8] Egenhofer, M., 1989. *Spatial Query Languages. PhD thesis, University of Maine*, Orono, ME, 1989.

[9] Egenhofer, M., 1990. *Extending SQL for Cartographic Display*. Cartography and Geographic Information Systems, 1990.

[10] Egenhofer, M., 1992. *Why not SQL!* International Journal of Geographical Information Systems 6(2), 71-85.

[11] Egenhofer, M., 1994. Spatial SQL: *A Query and Presentation Language*. IEEE Transactions on Knowledge and Data Engineering : 86–95, 1994.

[12] P. Aronson and S. Morehouse (1983). The ARC/INFO Map Library: *A Design for a Digital Geographic Database*. In: Auto-Carto VI, pages 346-382, Ottawa, 1983.

[13] M. Brodie, J. Mylopoulos, and J. Schmidt (1984). *On Conceptual Modelling, Perspectives from Artificial Intelligence, Databases, and Programming Languages*. Springer Verlag, New York, NY, 1984.

[14] Frank, A. U., 1982. Mapquery : *database query language for retrieval of geometric data and its graphical representation*. ACM SIG Computer Graphics 19(3), 199-207.

[15] Nurseitov, N., Paulson, M., 2009: *Comparison of JSON and XML data interchange formats: A case study*. In: CAINE Conf., pp. 157–162 (2009).

[16] Câmara, G., Palomo, D., Cartaxo Modesto de Souza, R. and Fradico de Oliveira O. R., 2005. *Towards a generalized map algebra: principles and data types*. In Proceedings of the 7th Brazilian Symposium on GeoInformatics - GEOINFO 2005, Campos do Jordão, Brazil, ISBN: 85-17-00022-6, November 2005.

[17] Guting, R. H., 1988. Geo-Relational Algebra: *A Model and Query Language for Geometric Database Systems*. Proceedings International Conference on Extending Database Technology, Venice, Italy, Springer-Verlag, pp. 506-527.

[18] Herring, J. R., 2006. *OpenGIS Implementation Specification for Geographic information – Simple feature access*. Common architecture, October 2006. Available from: www.opengeospatial.org/standards/sfs.

[19] Postgis (2003) : *Postgis, geographic objects for Postgres*, http://postgis.refractions.net, 2003

[20] M. Brodie. Association (1981): *A Database Abstraction for Semantic Modeling*. In: 2nd International Entity-Relationship Conference, Washington, D.C., 1981.