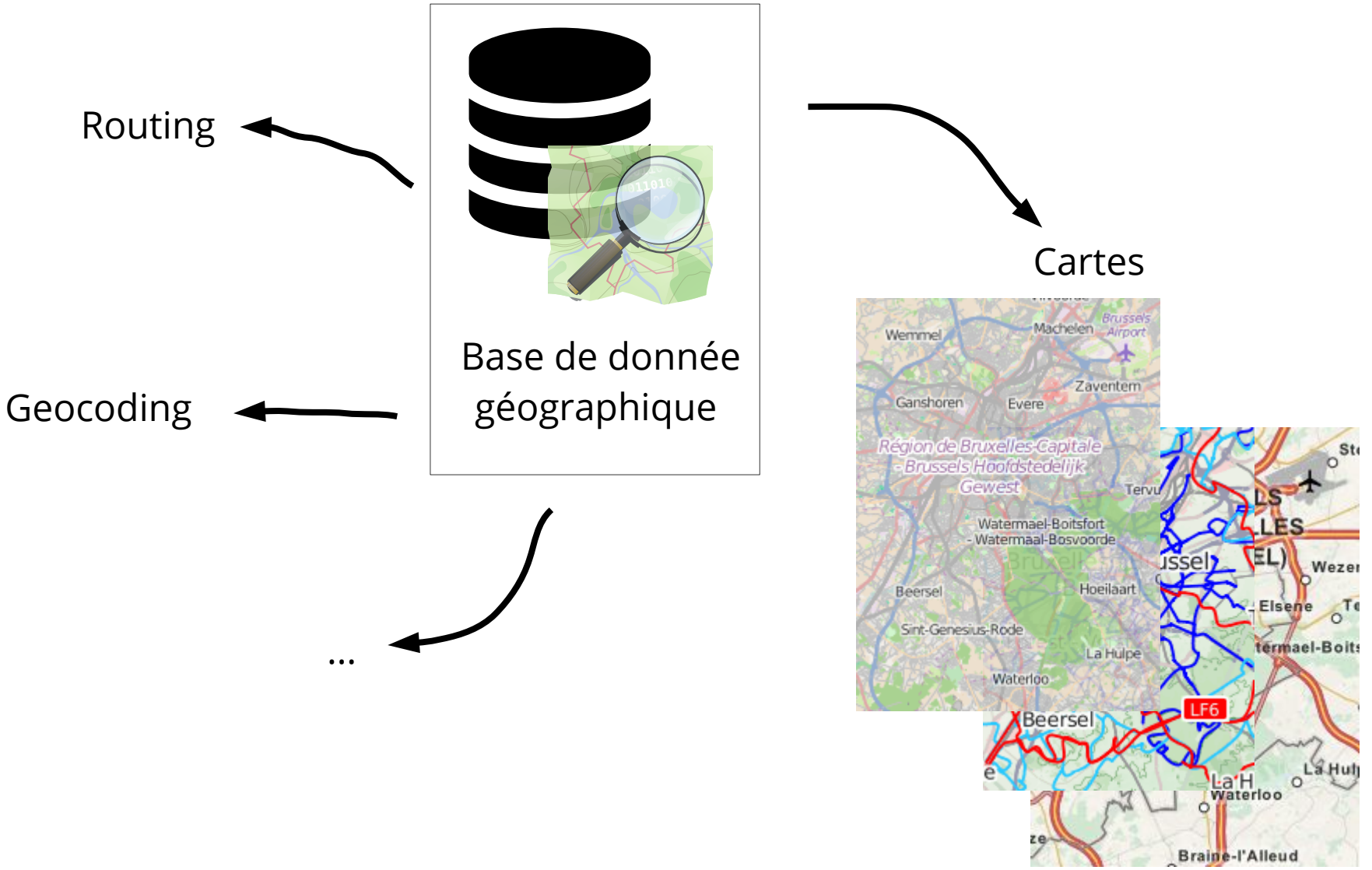
Overpass API

Requêtes Openstreetmap avec Overpass API

By Julien Fastré – julien@champs-libres.coop

UDOS – 10 juillet 2015



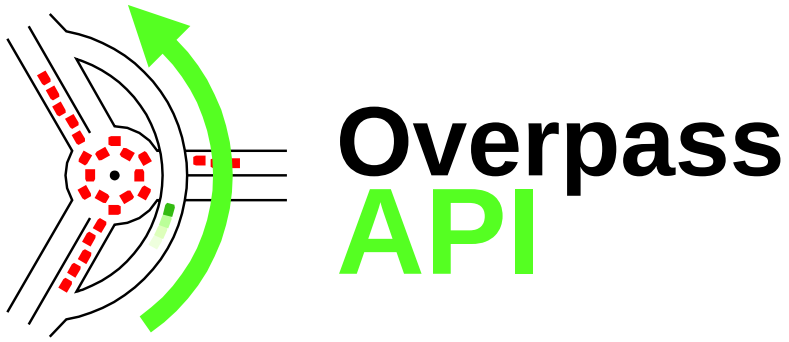
Rappel : OSM est une base de donnée géographique

Procédure d'extraction est complexe

Maintien des données extraits à jour => refaire la procédure d'extraction

```
<way id="89356456">
  <nd ref="1103964090"/>
  <nd ref="1036482478"/>
  <nd ref="1036482508"/>
  <nd ref="1103964478"/>
  <nd ref="1103964090"/>
  <tag k="amenity" v="bicycle_parking"/>
  <tag k="bicycle_parking" v="stands"/>
  <tag k="capacity" v="20"/>
  <tag k="covered" v="no"/>
  <tag k="fee" v="no"/>
  <tag k="parking" v="surface"/>
</way>
<node id="1036482478" lat="50.6410896" lon="5.5715259"/>
<node id="1036482508" lat="50.6410045" lon="5.5713088"/>
<node id="1103964090" lat="50.6411370" lon="5.5714818"/>
```

L'export des données a souvent été complexe



Logo
overpass
turbo

Overpass api & overpass turbo permettre d'exécuter des requêtes sur la BDD OSM

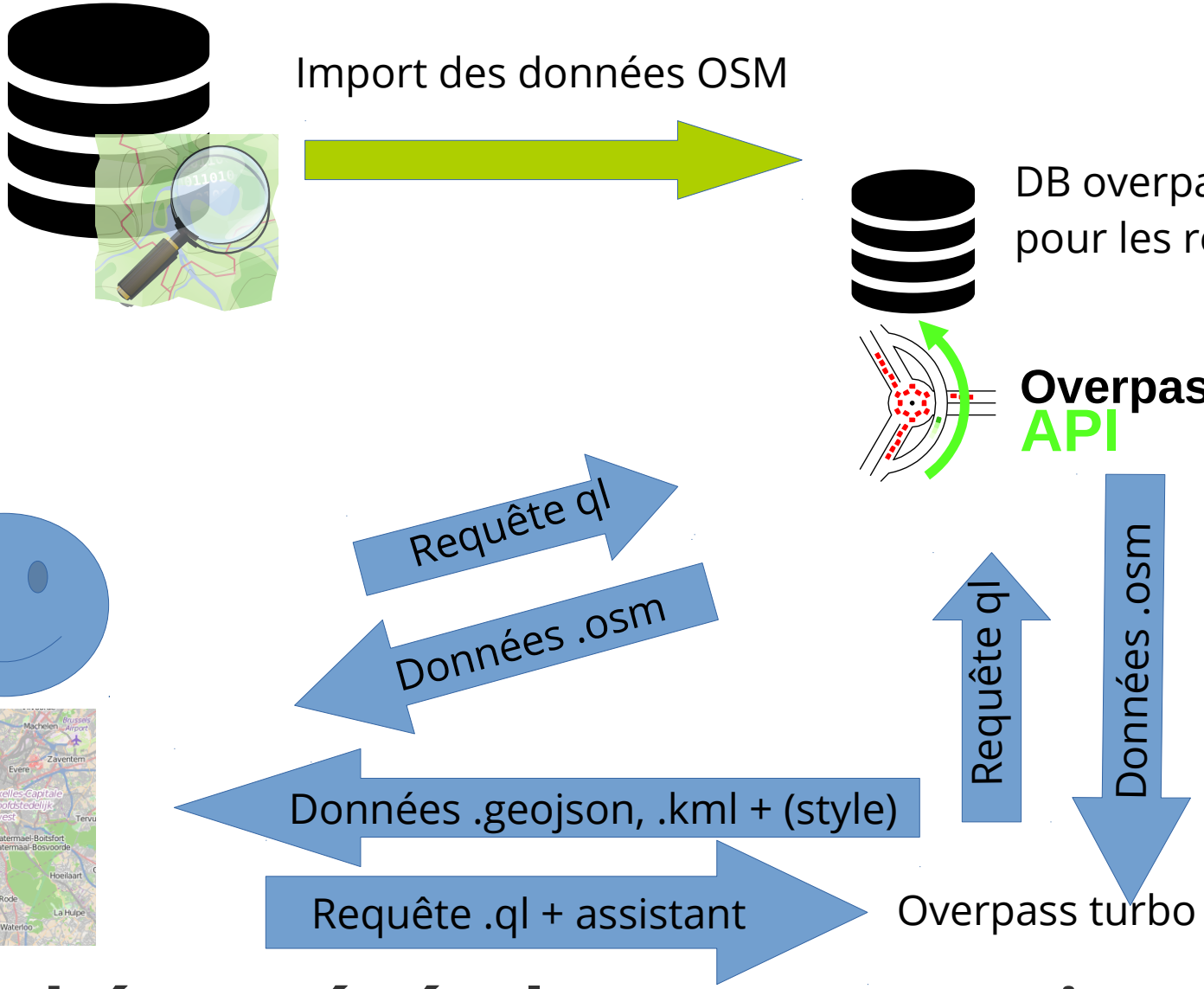


Schéma général – overpass api et overpass-turbo

- Syntaxe supplémentaire « {{ }} »
- Export des données en .geojson, .kml, ...
- Stylisation des données exportées en MapCSS
- Export & incorporation d'une carte dans une page web
- Interface plus conviviale
- Assistant de requête

  Peut se connecter à différentes instances overpass api

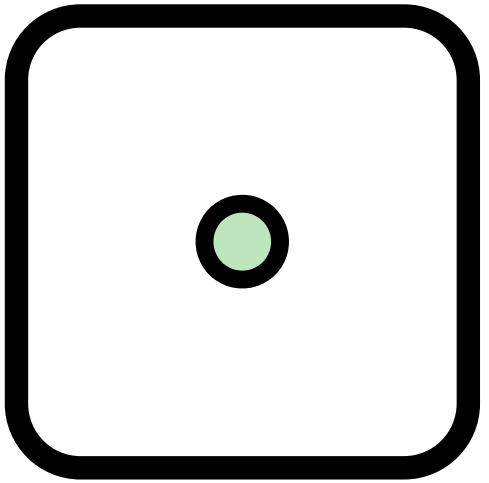
Overpass turbo ajoute des fonctionnalités

Trouver le code source ?

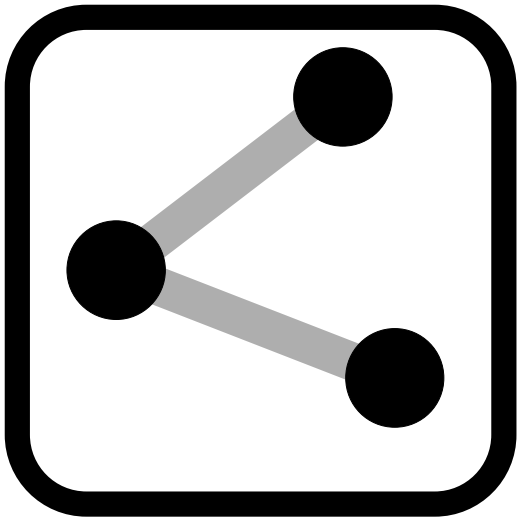
- Overpass api : https://github.com/drolbr/Overpass-API
- Overpass turbo : https://github.com/tyrasd/overpass-turbo

Overpass api est un logiciel libre

Modèle de données

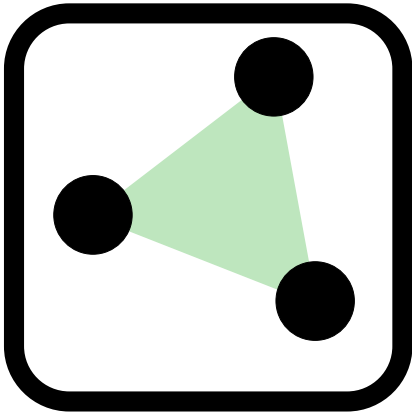


Au commencement était le point (node)…



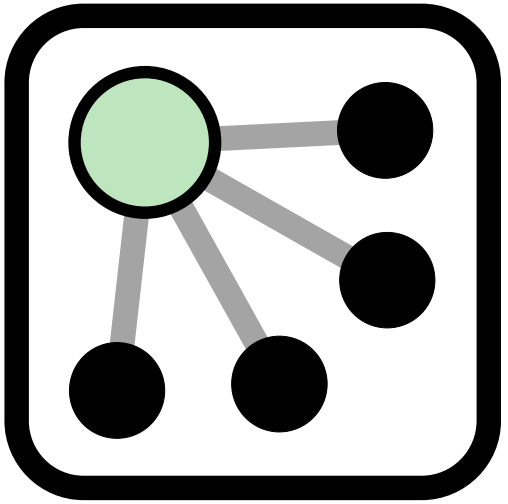
Points reliés et ordonnés

… puis est venue le chemin (way)



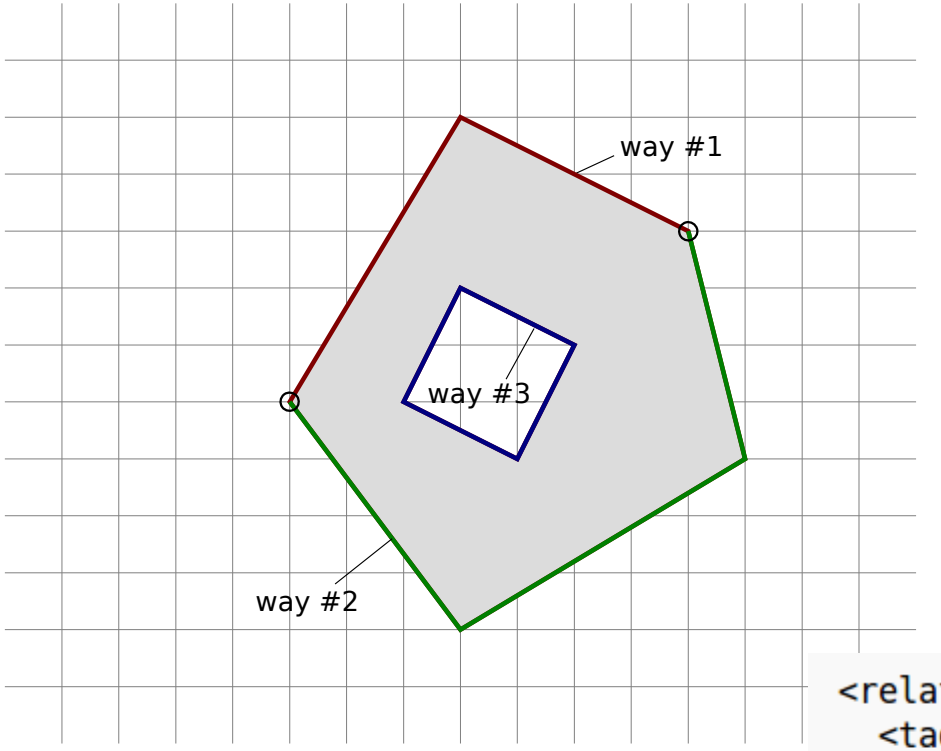
Dans un chemin, lorsque le dernier point est égal au premier, on peut supposer qu'il s'agit d'un polygone

Il n'y a pas de « polygones » dans le modèle original



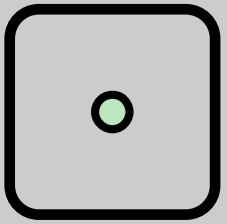
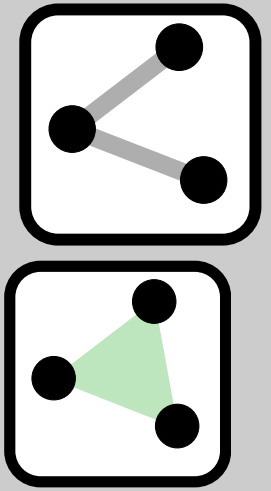
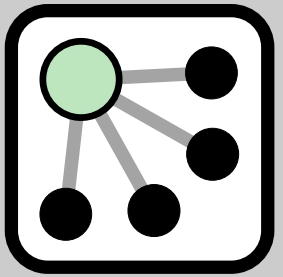
Permet de relier des éléments (way, node, ou autre relations) entre eux

Les relations : des liens entre éléments

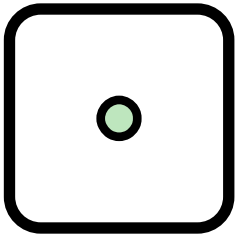


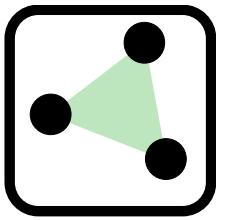

```
<relation id="1">
  <tag k="type" v="multipolygon" />
  <member type="way" id="1" role="outer" />
  <member type="way" id="2" role="outer" />
  <member type="way" id="3" role="inner" />
</relation>
```

Multipolygones : relations entre chemins

Noeud	Chemin	Relation	Area
			Dérivé des relations administratives
`node`	`way`	`rel`	`area`

Correspondance overpass api





amenity=bicycle_parking

bicycle_parking=stands
Capacity=8

operator=Ville de Bruxelles

covered=no

source=survey

Ajout de tags

Le langage QL

```
[out:json][timeout:25];

// cathédrale de digne les bains
way(112485950);

// print results
out tags;
```

```
// les éléments qui précèdent sont omis ici
{
  "type": "way",
  "id": 112485950,
  "tags": {
    "amenity": "place_of_worship",
    "building": "church",
    "denomination": "catholic",
    "heritage": "2",
    "heritage:operator": "mhs",
    "mhs:inscription_date": "1906-10-30",
    "name": "Cathédrale Saint-Jérôme",
    "ref:FR:CEF": "04070_09",
    "ref:mhs": "PA00080378",
    "religion": "christian",
  }
}
```

Par défaut, la sortie d'une instruction est placées dans la variable ._
Par défaut, l'entrée d'une instruction est cette variable ._

Sélection par id

```
//sélection d'un chemin
way(int);

// sélection d'un noeud
node(int) ;

//sélection d'une relation
rel(int) ;
```

Syntaxe sélection par id

La sortie

```
way(112485950);

out ids;
```

```
{
  "type": "way",
  "id": 112485950
}
```

Niveaux de verbosité de sortie : ids

```
way(112485950);

out skel;
```


```
{
  "type": "way",
  "id": 112485950,
  "nodes": [
    1278398772,
    1278402155,
    1278316961,
    1278384839,
    1278298824,
    1278329951,
    1278397645,
    ...
   1278398772
  ]
}
```


Niveaux de verbosité : skel

```
way(112485950);

out tags;
```


```
{
 "type": "way",
 "id": 112485950,
 "tags": {
  "amenity": "place_of_worship",
  "building": "church",
  "denomination": "catholic",
  "heritage": "2",
  "heritage:operator": "mhs",
  "mhs:inscription_date": "1906-10-30",
  "name": "Cathédrale Saint-Jérôme",
  "ref:FR:CEF": "04070_09",
  "ref:mhs": "PA00080378",
  "religion": "christian",
  "source": "cadastre-dgi-fr source : Direction G
2011"
 }
}
```


Niveaux de verbosité : tags

```
[out:json];

way(112485950);

out body;
```

Ajout les tags du niveau « tags » et les noeuds nécessaires à l'affichage du niveau « skel ».

Niveaux de verbosité : body

```
way(112485950);

out meta;
```


```
{
 "type": "way",
 "id": 112485950,
 "timestamp": "2015-04-12T11:48:10Z",
 "version": 5,
 "changeset": 30161571,
 "user": "MIchel SDIS 04",
 "uid": 664020,
 "nodes": [
  1278398772,
  ...
  1278398772
 ],
 "tags": {
  "amenity": "place_of_worship",
  "building": "church",
  "denomination": "catholic",
  "heritage": "2",
  "heritage:operator": "mhs",
  "mhs:inscription_date": "1906-10-30",
  "name": "Cathédrale Saint-Jérôme",
  "ref:FR:CEF": "04070_09",
  "ref:mhs": "PA00080378",
  "religion": "christian",
  "source": "cadastre-dgi-fr source : Direction Générale des Impôts - Cadastre. Mise à jour : 2011"
 }
}
```


Niveaux de verbosité : meta

```
node(1278398772);
out skel;
```


```
{
  "type": "node",
  "id": 1278398772,
  "lat": 44.0922110,
  "lon": 6.2356800
}
```


Pour afficher un noeud, ses coordonnées son nécessaires

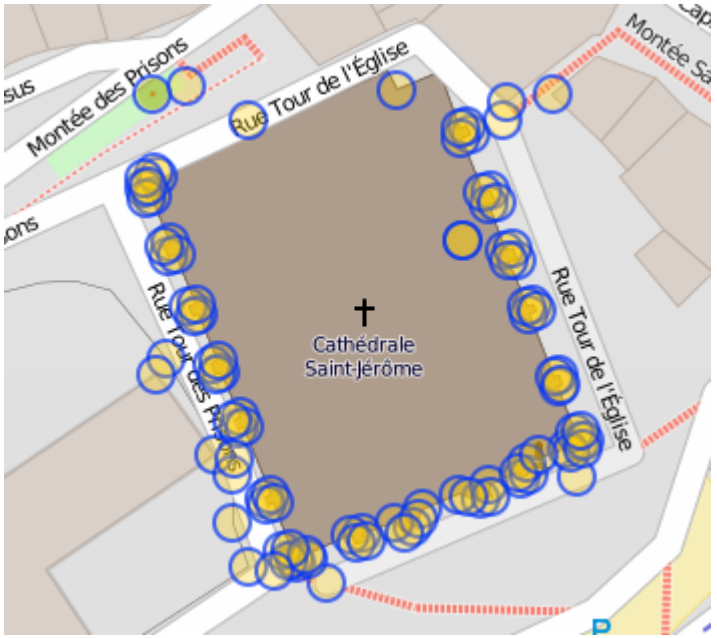
```
way(112485950);
out skel;
>; // afficher les enfants du chemin
out skel;
```


```
{
  "type": "way",
  "id": 112485950,
  "nodes": [
    1278398772,
    ...
    1278398772
  ]
},
{
  "type": "node",
  "id": 1278277236,
  "lat": 44.0919690,
  "lon": 6.2358050
},
  // ...
```


Pour afficher d'un polygone, ses noeuds doivent faire partie du résultats

Les filtres

```
node(44.0918, 6.2356790, 44.09231, 6.2362690);
out skel;
```




```
{
  "type": "node",
  "id": 353287988,
  "lat": 44.0922655,
  "lon": 6.2361603
},
{
  "type": "node",
  "id": 577253960,
  "lat": 44.0918219,
  "lon": 6.2359250
},
```


Filtre par BBOX

way["highway"](44.091, 6.2356, 44.0923, 6.23626);

out skel;

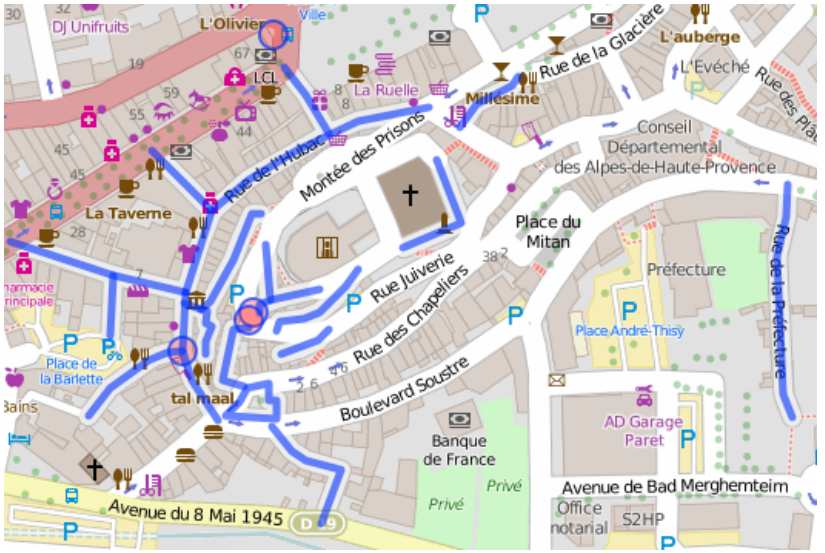

```
{
  "type": "way",
  "id": 31573271,
  "nodes": [
    354511050,
  ],
  "tags": {
    "highway": "unclassified",
    "name": "Rue Tour de l'Église"
  }
},
{
  "type": "way",
  "id": 31573274,
  "nodes": [
    353287998,
  ],
  "tags": {
    "foot": "yes",
    "highway": "pedestrian",
    "name": "Rue du Figuier"
  }
},
```


Filtrer par tag clé/valeur

```
//équivalent
way[highway=pedestrian];
way["highway"="pedestrian"];

// chemins qui n'ont *pas* le tag pedestrian
way[highway!=pedestrian];
// (sélectionne aussi ceux qui ne sont pas des highway)

// sélectionne les chemins qui ont le tag highway
way[highway];

// sélection selon expression régulière
way[name~"^Rue"]; // tout ce qui commence par "Rue"
```

Filtrer par clé/valeur

```
//l'area de "dignes les bains" est recherchée par Nominatim
//et stockée dans la variable "dignesArea"
{{geocodeArea:"Digne-Les-Bains,France"}}->.dignesArea;

//on recherche tous les chemins piétons à l'intérieur de l'aire
// de "digne les bains"
way["highway"="pedestrian"](area.dignesArea);

out body;
>;
out skel;
```

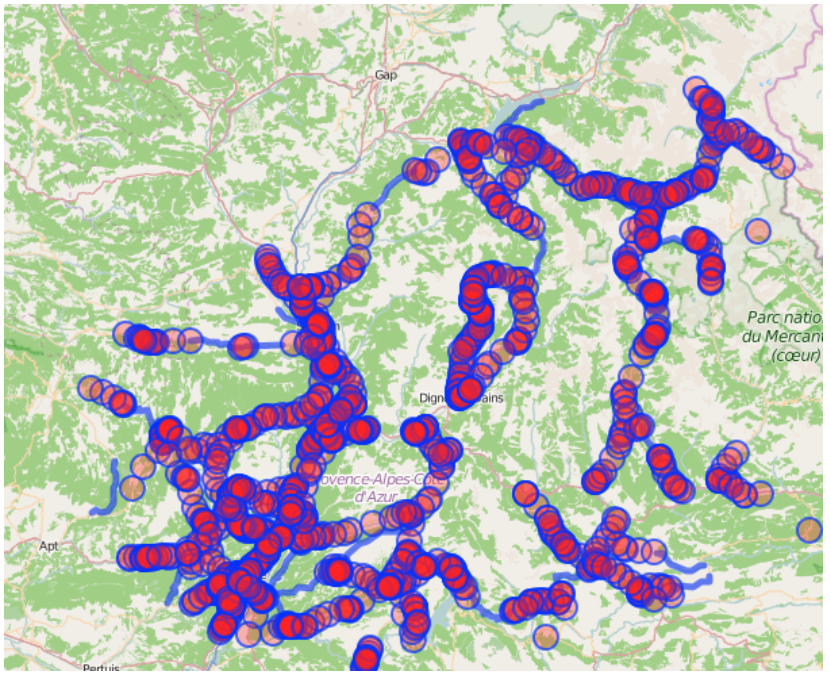


Filtre par « area » (+ variables)

```
[out:json][timeout:25];

//trouver l'area "Alpes-de-haute-Provence"
{{geocodeArea:"Alpes-de-Haute-Provence"}}->.dept;

//filtrer les routes principales
way["highway"="primary"](area.dept)->.routes;

//on applique un nouveau filtre sur la variable .routes
//filtrer les routes dont la ref. commence par D
way.routes["ref"~"^D"];

out body;
>;
out skel qt;
```

Combiner des filtres

```
[out:json][timeout:25];

//trouver l'area "Alpes-de-haute-Provence"
{{geocodeArea:"Alpes-de-Haute-Provence"}}->.dept;

//trouver les routes
(
  way["highway"="primary"](area.dept);
  way["highway"="secondary"](area.dept);
)->.routes;

//filtrer les routes dont la ref. commence par D
way.routes["ref"~"^D"];

out body;
>;
out skel qt;
```



Jointures (requêtes « UNION »)

- Trouver les amenities qui sont accessibles au personnes en chaise roulante à Digne-Les-Bains ;
  - wheelchair=yes
  - Amenity=*

  ! les amenities peuvent être des points ou des polygones !
- Trouver les routes avec des aménagements cyclables à Liège, Belgium ;
  - highway=cycleway
  - cycleway=*
  - cycleway:left=*
  - cycleway:right =*
  - bicycle:oneway=no
  - oneway:bicycle=no
- Trouver les banques avec distributeurs de billet
  - amenity=bank
  - atm=yes

À vous !

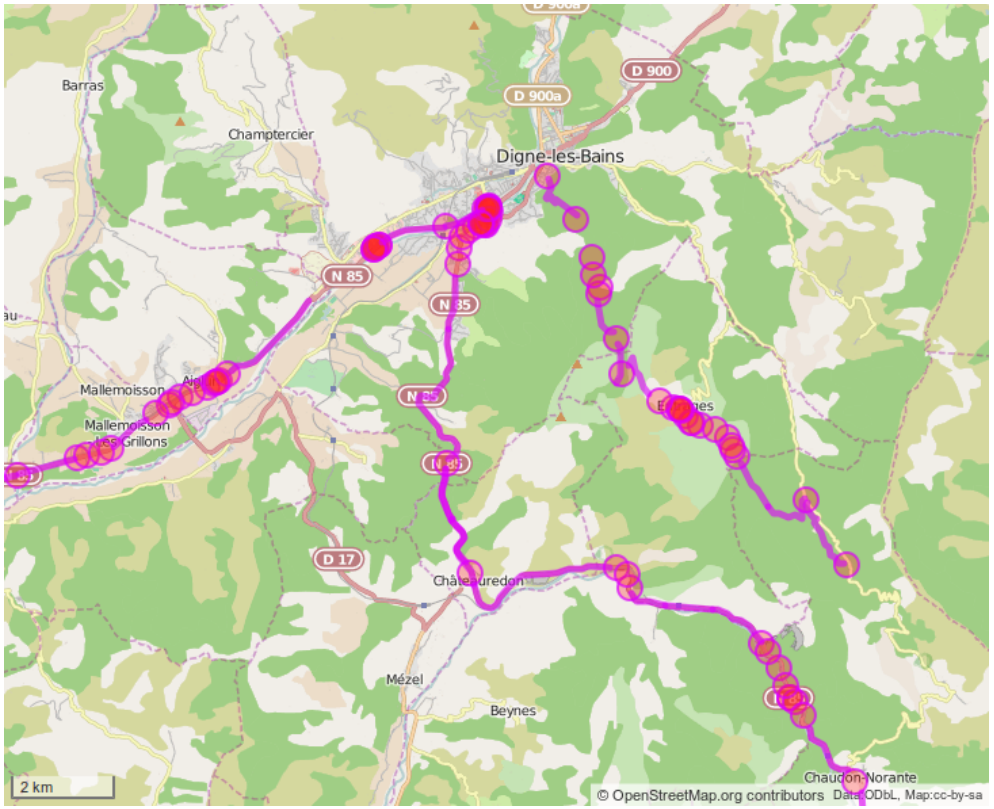
```
// renvoie tous les éléments qui sont présent dans .a
// et absent de .b
(.a - .b) ;
```

Différence

>	<
Recurse down	Recurse up
• Tous les noeuds d'un chemin ; • Tous les membres d'une relation (jusqu'au noeud) ; •	• Tous les relations dont fait partie un chemin ; • Tous les chemins dont fait partie un noeud ;
>>	<<
Idem, ajoute les relations qui seraient membres de la relation	Idem, ajoute les relations dont ferait partie les relations

Récursions

```
[out:json][timeout:25];

//rel(3644265);
rel["name"="Route Napoléon Historique"];
out meta;
> ;
out skel qt;
```



Exemple Récursion sur une relation

```
way(112485950);
out skel;
>; // afficher les enfants du chemin
out skel;
```


```
{
  "type": "way",
  "id": 112485950,
  "nodes": [
    1278398772,
    ...
    1278398772
  ]
},
{
  "type": "node",
  "id": 1278277236,
  "lat": 44.0919690,
  "lon": 6.2358050
},
  // ...
```

Récursion sur un chemin

```
[out:json][timeout:25];

//trouver digne les bains
{{geocodeArea:"Digne-Les-Bains"}}->.digne;

// sélectionner toutes les rues qui contiennnt "Simone-Pellissier"
way["name"~"Simone-Pellissier"](area.digne)->.avenue;

//trouver les parents de l'avenue
.avenue<->.parents;

.parents out tags;
```

Récursion « up »

```
{
  "type": "relation",
  "id": 2880632,
  "tags": {
    "colour": "#F89C1C",
    "name": "LER Marseille / Nice / Sisteron / Gre
    "network": "Région Provence-Alpes-Côte-d'Az
    "operator": "SCAL, Phocéens Cars",
    "public_transport:version": "1",
    "ref": "LER 31",
    "route": "bus",
    "type": "route"
  }
}
```

Filtrer récursivement un type d'élément

Ex. : `node(w._)  ;`

w => way
r => relation
n => node
+b : backward

```
node(w);      // sélectionner les noeuds enfants d'une entrée
node(r);      // sélectionner les noeuds membres de la relation d'entrée
way(bn);      // sélectionner les chemins parents des noeuds de l'entrée
way(r);       // sélectionner les chemins membres des relations de l'entrée
rel(bn);      // sélectionner les relations qui comptent des noeuds membres parmi l'entrée
rel(bw);      // sélectionner les relations qui comptent des chemins membres parmi l'entrée
rel(r);       // selectionner tous les membres de type relation qui sont membre de l'entrée
rel(br);      // selectionner les relations parentes de toutes le relations présentents dans l'entrée
```

Filtres de récursion

Trouver tous les chemins qui font partie d'une ligne de bus à Digne-Les-Bains.

- Ligne de bus : relation, de `type=route` & `route=bus`

Trouver toutes les rues de Liège qui croisent une rue avec un aménagement cyclable mais qui n'ont pas d'aménagement cyclable.

A vous

```
[out:json][timeout:25];

//cathédrale de digne
way(112485950);

// noeuds à 50m de la cathédrale
node(around:50);

out skel;
```

Around

- Trouver les amenities à moins de 50m d'un parking vélo

  Amenity=*

  amenity=bicycle_parking

- ...

A vous !

Overpass turbo permet d'exporter en :

- Geojson
- Données brutes
- KML
- Export direct depuis overpass api (pour éviter l'UI de overpass turbo)

Export des données

- Requêtes QL :

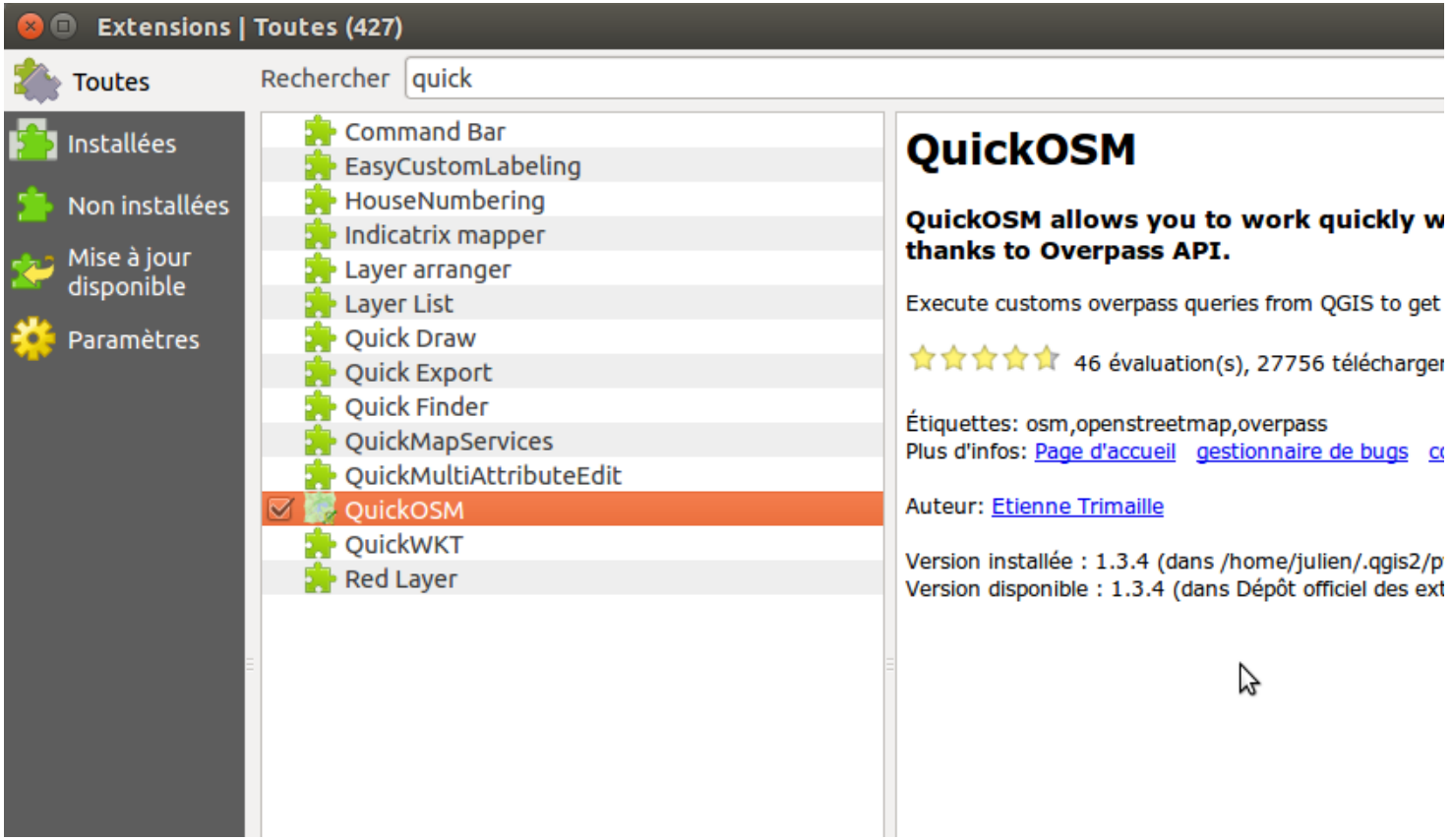
  Utilisé jusqu'ici...

- Requêtes Overpass-XML

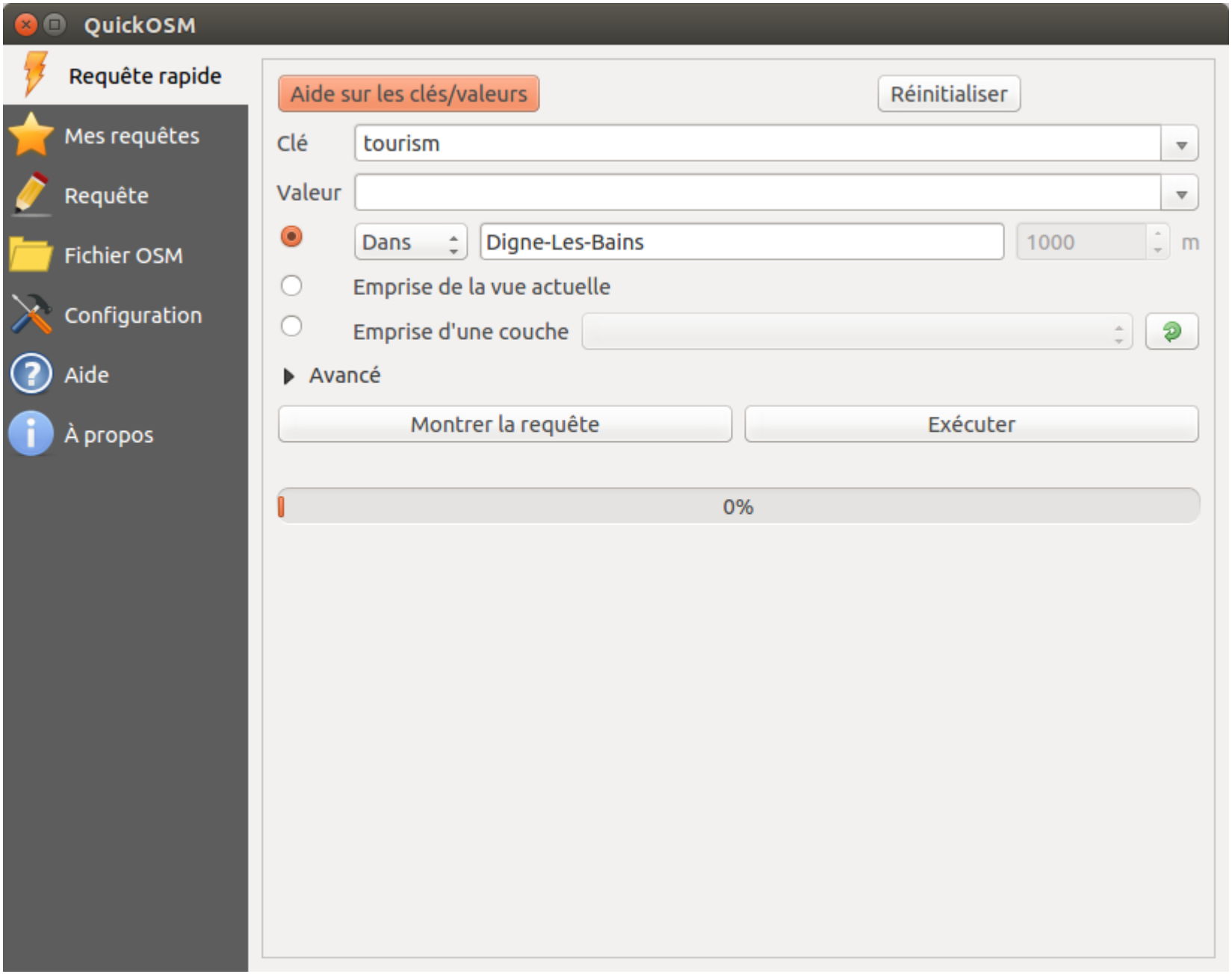
```
<osm-script output="json" timeout="25">
  <id-query into="digne" ref="3600085909" type="area"/>
  <query into="_" type="node">
    <area-query from="digne" into="_" ref=""/>
    <has-kv k="amenity" modv="" v="bicycle_parking"/>
  </query>
  <print e="" from="_" geometry="skeleton" limit=""
mode="skeleton" n="" order="quadtile" s="" w=""/>
</osm-script>
```

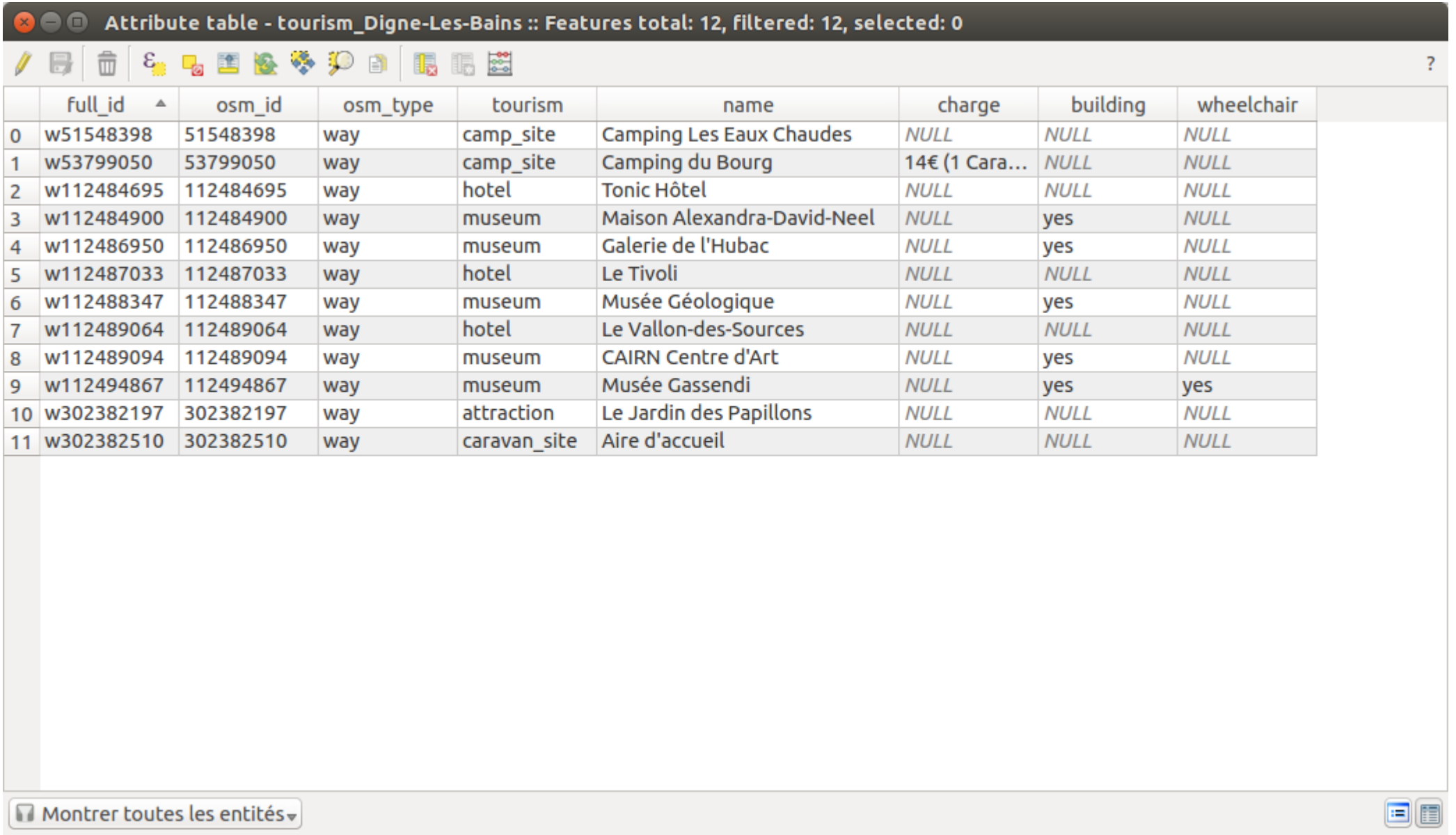
Différents formats de requêtes

Le plugin QuickOSM permet d'importer des requêtes Overpass API dans une couche QGIS



Utilisation de OverpassAPI depuis QGIS


QuickOSM
Requête rapide
Mes requêtes
Requête
Fichier OSM
Configuration
Aide
À propos
Aide sur les clés/valeurs
Réinitialiser
Clé
tourism
Valeur
Dans
Digne-Les-Bains
1000
m
Emprise de la vue actuelle
Emprise d'une couche
Avancé
Montrer la requête
Exécuter
0%



	full_id	osm_id	osm_type	tourism	name	charge	building	wheelchair
0	w51548398	51548398	way	camp_site	Camping Les Eaux Chaudes	*NULL*	*NULL*	*NULL*
1	w53799050	53799050	way	camp_site	Camping du Bourg	14€ (1 Cara…	*NULL*	*NULL*
2	w112484695	112484695	way	hotel	Tonic Hôtel	*NULL*	*NULL*	*NULL*
3	w112484900	112484900	way	museum	Maison Alexandra-David-Neel	*NULL*	yes	*NULL*
4	w112486950	112486950	way	museum	Galerie de l'Hubac	*NULL*	yes	*NULL*
5	w112487033	112487033	way	hotel	Le Tivoli	*NULL*	*NULL*	*NULL*
6	w112488347	112488347	way	museum	Musée Géologique	*NULL*	yes	*NULL*
7	w112489064	112489064	way	hotel	Le Vallon-des-Sources	*NULL*	*NULL*	*NULL*
8	w112489094	112489094	way	museum	CAIRN Centre d'Art	*NULL*	yes	*NULL*
9	w112494867	112494867	way	museum	Musée Gassendi	*NULL*	yes	yes
10	w302382197	302382197	way	attraction	Le Jardin des Papillons	*NULL*	*NULL*	*NULL*
11	w302382510	302382510	way	caravan_site	Aire d'accueil	*NULL*	*NULL*	*NULL*

```
wget
http://overpass-api.de/api/interpreter?data=
%5Bout%3Ajson%5D%5Btimeout%3A25%5D%3Barea%28
3600085909%29%2D%3E%2Edigne%3Bnode%28area%2E
digne%29%5B%22amenity%22%3D%22bicycle%5Fpark
ing%22%5D%3Bout%20skel%20qt%3B%0A
```

- Overpass-api permet d'obtenir la requête échappée

Utiliser Overpass-api en ligne de commande

- Pour pouvoir se connecter directement à Overpass-Api : utiliser en-tête CORS
- Relativement déconseillé :

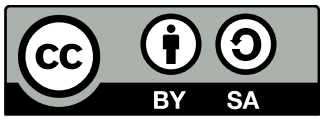
    Plutôt mettre en place un script cron

Utiliser Overpass API dans OpenLayers/Leaflet

La documentation complète de QL est disponible à l'adresse http://wiki.openstreetmap.org/wiki/Overpass_API/Overpass_QL

Toute la doc...

Merci, bon retour !



Contact : julien@champs-libres.coop