Université IBM i

7 novembre 2023

IBM Innovation Studio Paris

S01 – Intégrez les fonctions géospatiales de DB2 dans vos applications

11:15 - 12:15

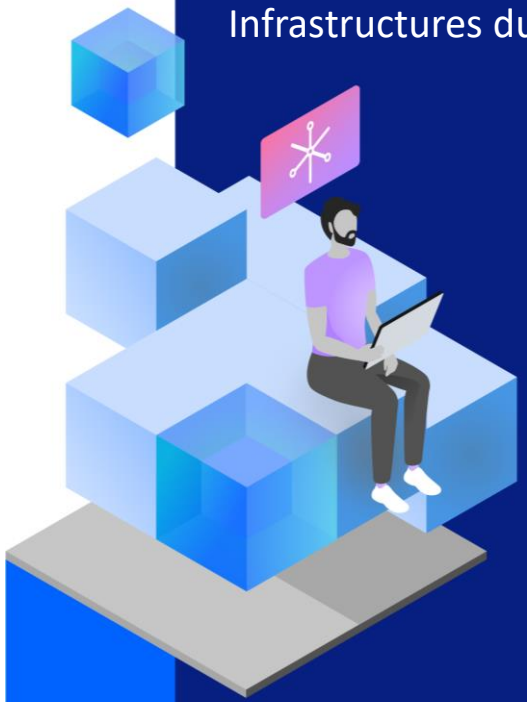
Philippe Bourgeois
IBM France
pbourgeois@fr.ibm.com

infrasdufutur

#ibmi
#uii2023
#infrastructuredufuturIBM23

IBM

Infrastructures du futur



7 et 8 novembre 2023

Agenda

- 1. Analytique géospatiale : vocabulaire
- 2. Analytique géospatiale : pourquoi et comment ?
- 3. DB2 for i : données et fonctions géospatiales
- 4. Pour aller plus loin

1. Analytique géospatiale : vocabulaire

Vocabulaire – Entité géographique

- **1. Entité géographique**
  - Elément du monde réel dont l'emplacement est identifiable :
    - Un objet
      - Un fleuve, une forêt, une vallée, un lac, une chaîne de montagnes
      - Un bâtiment, un barrage, un pont, une route, une ville
      - …
    - Un espace
      - Une zone de sécurité autour d'un site dangereux
      - La zone de commercialisation desservie par une entreprise
      - …
    - Le lieu d'un événement
      - L'intersection où est survenu un accident de voiture
      - Le magasin dans lequel a eu lieu un achat
      - …

Vocabulaire – Informations géospatiales

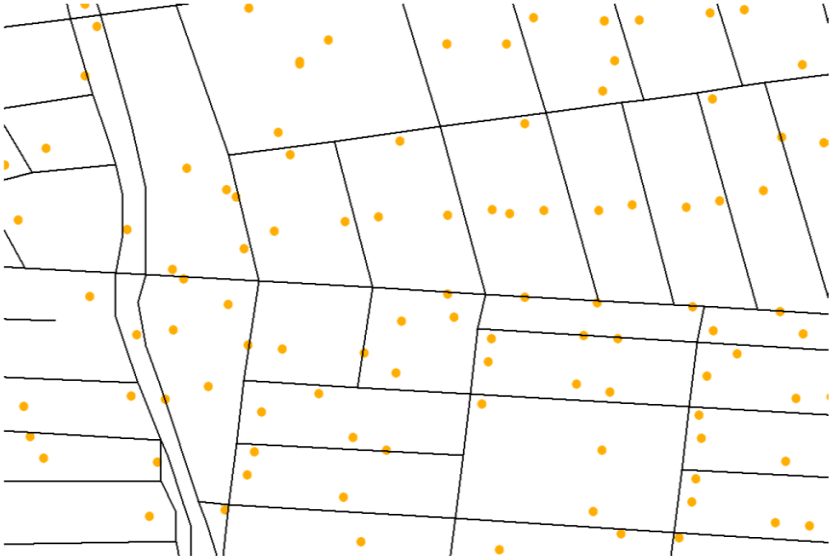
- **2. Informations géospatiales**
  - L'emplacement des *entités géographiques*
    - Par exemple, la longitude et la latitude d'une ville
  - L'emplacement des *entités géographiques* les unes par rapport aux autres
    - Par exemple, l'emplacement de tous les hôpitaux et cliniques d'une ville ou la proximité des résidents d'une ville à une zone sismique particulière
  - Les façons dont les *entités géographiques* sont reliées les unes aux autres
    - Par exemple, l'information selon laquelle un pont est contenu dans une zone donnée
  - Les mesures qui s'appliquent à une ou plusieurs *entités géographiques*
    - Par exemple, la distance entre un immeuble de bureaux et la limite de son terrain, ou la longueur du périmètre d'une réserve naturelle

Vocabulaire – Données géospatiales - Géométries

- **3. Données géospatiales** = emplacement des *entités géographiques* = coordonnées
  - Coordonnées qui identifient un objet
  - Coordonnées qui définissent un chemin
  - Coordonnées qui délimitent une frontière

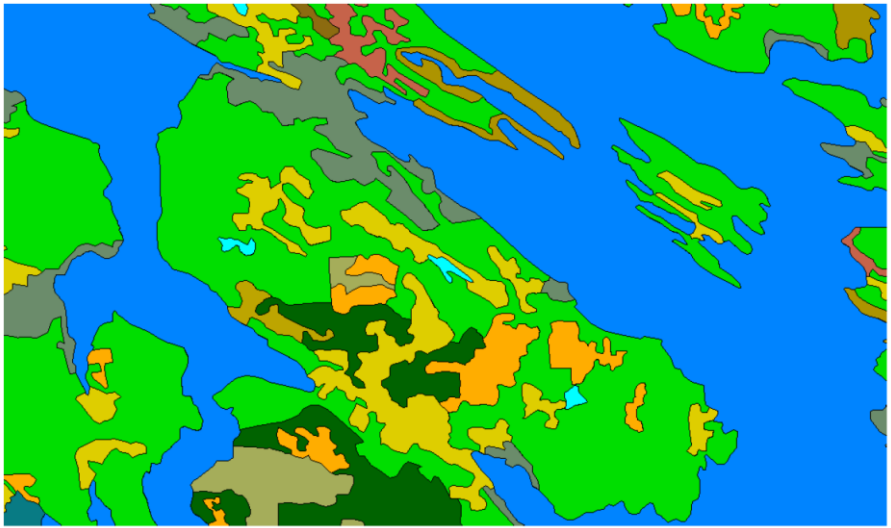
- **4. Géométrie** = représentation "technique" des *entités géographiques*
  - Points, lignes, polygones
  - Ensembles de points, lignes, polygones

Géométries – Points, lignes, polygones



- Les boîtes aux lettres sont représentées par des **points**
- Les rues sont représentées par des **lignes**

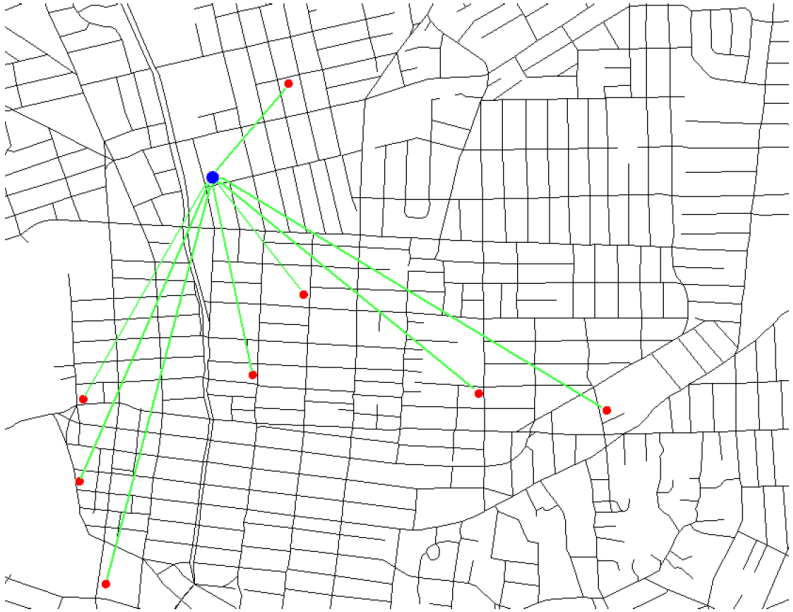
- Les classifications d'utilisation des sols sont représentées par des **polygones**



Vocabulaire – Analytique géospatiale

- **5. Analytique géospatiale**
  - a. Création et stockage de *données géospatiales* dans une base de données
  - b. Ensemble de fonctions permettant :
    - D'examiner et d'interpréter les *données géospatiales*
    - De comparer des *géométries* pour en tirer des enseignements
      - Obtenir les "relations" entre ces *géométries* :
        - Proximité
        - Contiguïté, connectivité
        - Recouvrement
    - De créer de nouvelles *géométries* :
      - Zone autour d'une *géométrie* (appelée buffer / tampon)
      - Union, intersection, différence de plusieurs *géométries*

Comparaison de géométries – Relations



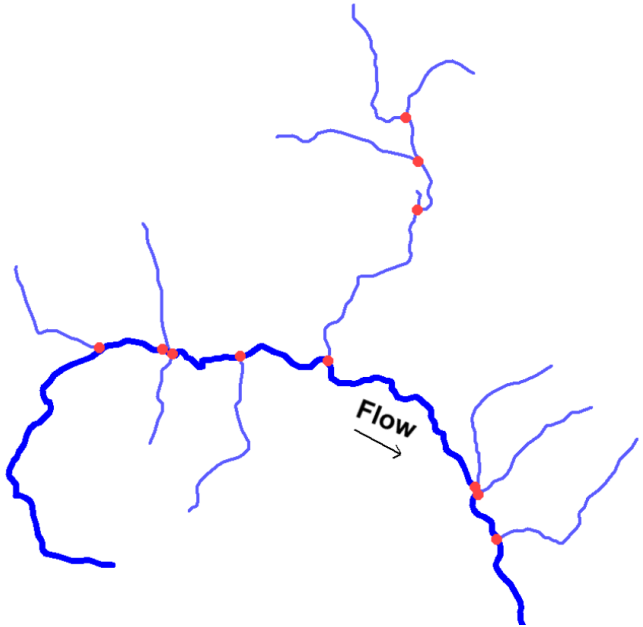
- **Distance** vers des agences, des points de vente…

- **Distance** entre une décharge de déchets toxiques et une parcelle de terrain que vous souhaitez acquérir

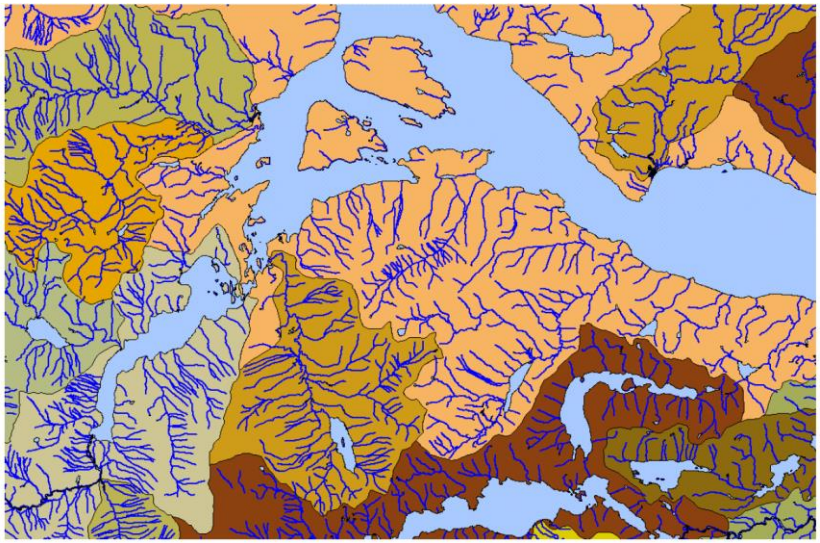
Comparaison de géométries – Relations

- Toutes les parcelles **contigües** à une parcelle donnée

- Les **connexions** des affluents sur une rivière

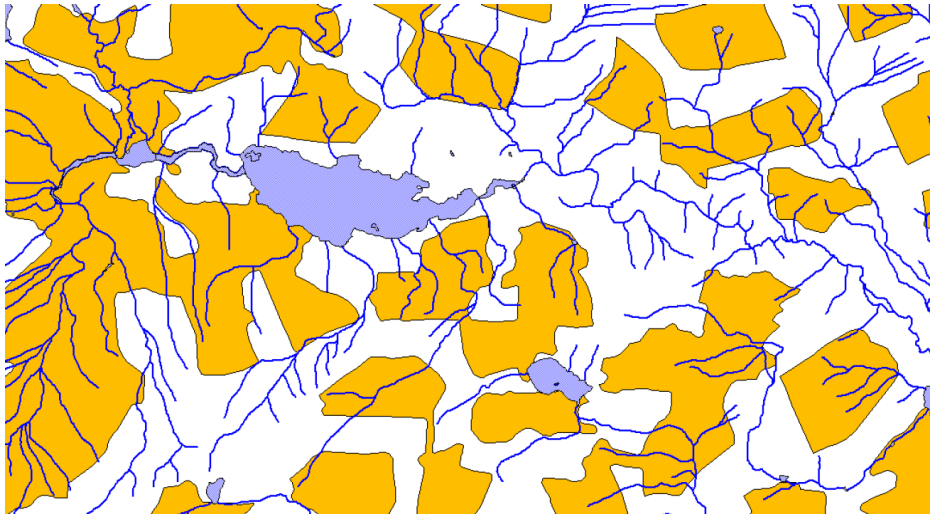


Comparaison de géométries – Relations

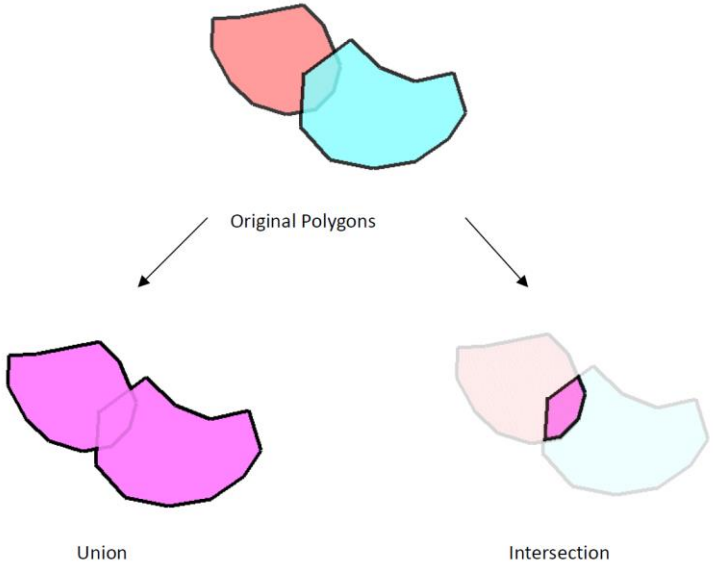


- Les cours d'eau **contenus** dans un territoire
- Les îles **contenues** dans un plan d'eau

- L'exploitation forestière en bordure de cours d'eau : forêts **contigües** aux plans d'eau et/ou qui **contiennent** des cours d'eau

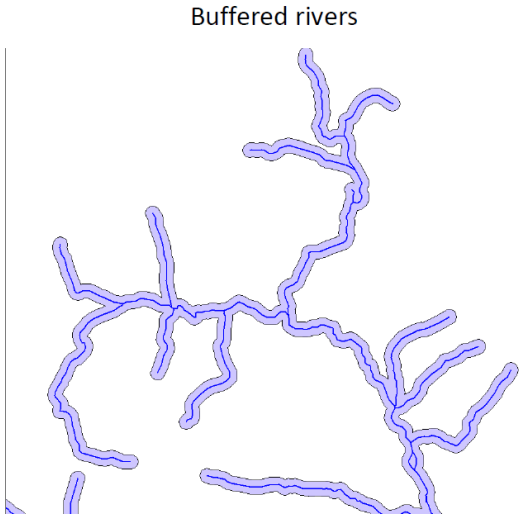


Création de nouvelles géométries

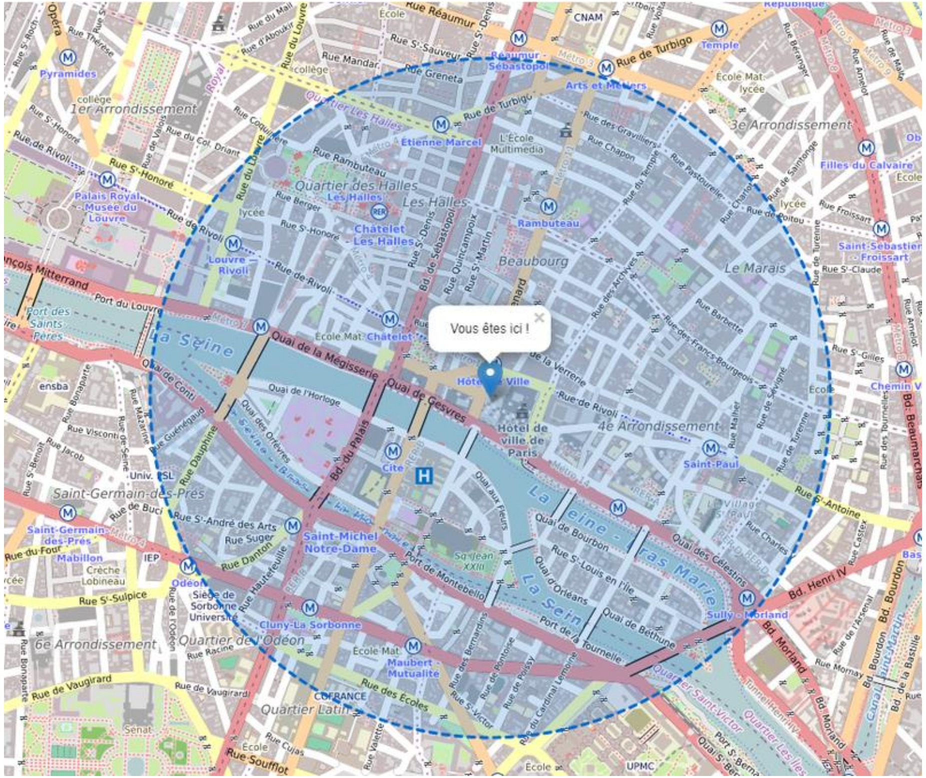


- **Union** / **intersection** de 2 parcelles

- Zones réservées de 2m de large en bordure de rivière (zones **tampon**)



Création de nouvelles géométries



- La zone de sortie autorisée lors d'un confinement (ici rayon de 1 km) (zone **tampon**)

2. Analytique géospatiale : pourquoi et comment ?

La plupart des sociétés disposent déjà d'informations "géographiques"

- Emplacement des magasins, clients, fournisseurs, dépôts, agences…
- Informations de suivi des transports
- Informations démographiques, statistiques, épidémiologiques…
- Informations sur les ressources naturelles
- Informations d'urbanisme, d'environnement, de propriété foncière…

Pourquoi faire de l'analytique géospatiale ?

- **Pour pouvoir répondre à ce type de questions :**
  - Où se trouvent nos magasins par rapport à nos clients ? Où se trouvent les magasins de nos concurrents ?
  - Où sont situés nos actifs ? Où se trouvent nos navires, conteneurs et camions ? Quelle est la route optimale ?
  - Où devrions-nous construire de nouvelles agences, franchises, magasins ?
  - Où devrions-nous faire de la publicité ?
  - Où se situent les risques météorologiques historiques et actuels ?
  - Quels sont nos clients qui sont en zone inondable ? . . .

Comment faire de l'analyse géospatiale ?

- Analytique géospatiale =
  - **Données** géospatiales : coordonnées de points, lignes, polygones

    +
  - **Fonctions** géospatiales : fonctions de comparaison et création de géométries

- Cela nécessite une **base de données** où l'on stockera les données géospatiales et qui permettra ensuite :
  - D'établir facilement, par SQL, des relations entre les géométries et créer de nouvelles géométries
  - D'établir facilement, par SQL, des relations avec d'autres données (non géospatiales) de la base
  - D'exploiter ces données directement par des interfaces Web, APIs, outils de BI et AI…

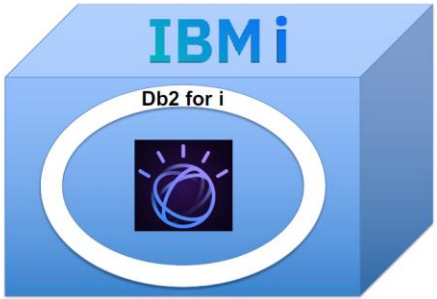
Quelle base de données ?

- Deux possibilités :
  - **1.** Bases de données *spécifiques* dites **"spatiales"** (SDBMS)
    - ESRI ArcSDE (couche au dessus de la plupart des bases de données, mais pas DB2 for i)
    - IBM DB2 Spatial Extender (mais pas de support de DB2 for i)
    - Oracle Spatial
    - Informix Special DataBlade
    - Geomedia (MS Access)
    - PostGIS (PostgreSQL)
    - Spatial Lite (SQLite)
    - …

Quelle base de données ?

- Deux possibilités :
  - **2. Intégration** dans les bases de données **relationnelles** existantes
    - **DB2 for i**
    - SQL Server …

- Intégration dans **DB2 for i**
  - A partir de la 7.4 TR7 ou la 7.5 TR1 - Décembre 2022
  - Technologie d'IA (IBM Watson) embarquée dans DB2 for i
  - Pas d'appel de services extérieurs
  - **Pas de données qui sortent de l'IBM i**
  - Pas de coût financier supplémentaire



3. DB2 for i : données et fonctions géospatiales

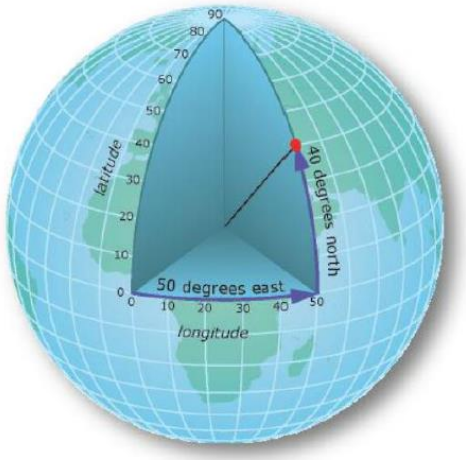
Analytique géospatiale et DB2 for i

- **Prérequis**
  - 7.4 : SF99704 niveau 23 minimum (25 pour les dernières nouveautés 2023)
  - 7.5 : SF99950 niveau 3 minimum (4 pour les dernières nouveautés 2023)

- **Fonctionnalités**
  - 8 nouveaux types de données
  - 51 nouvelles fonctions
  - 3 nouvelles vues dans le catalogue

- **Conformité**
  - Open Geospatial Consortium (OGC)
  - SQL/MM Spatial

Données géospatiales



- Données géospatiales :
  - Points
  - Lignes
  - Polygones

- Utilisation de **coordonnées** pour identifier une donnée géospatiale :
  - Pour un point :
    - X et Y (abscisse / ordonnée)
    - Utilisation du géocodage : **longitude** et **latitude** (**X** = longitude et **Y** = latitude)
  - Pour une ligne
    - Ensemble des coordonnées des différents points de la ligne
  - Pour un polygone
    - Ensemble des coordonnées des différents points du polygone

Système de coordonnées

- Système de coordonnées utilisé dans DB2 for i : **CGS WGS 1984**
  - Celui utilisé par les GPS
  - Unités de mesure : mètres et degrés

- Pour plus d'informations : visualiser le contenu de la vue **QSYS2.ST_COORDINATE_SYSTEMS** du catalogue DB2

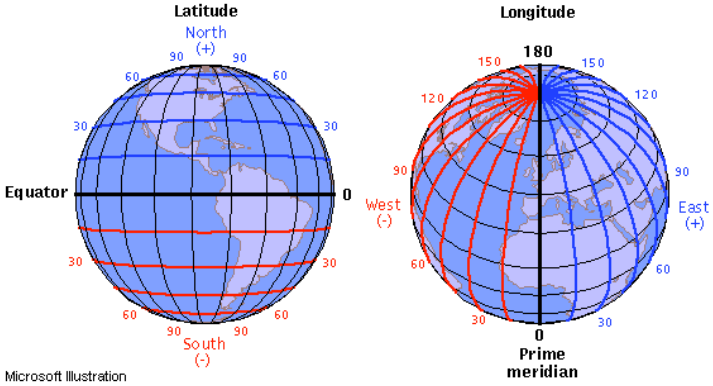
```
9  SELECT * FROM QSYS2.ST_COORDINATE_SYSTEMS;
```

COORDSYS_NAME	COORDSYS_TYPE	DEFINITION
GCS_WGS_1984	GEOGRAPHIC	GEOGCS[WGS 84,DATUM[WGS_1984,SPHEROID[WGS 84,6378137,298.257223563,AUTHORITY[EPSG,7030]],AUTHORITY[EPSG,6326]],PRIMEM[Greenwich,0,AUTHORITY[EPSG,8901]],UNIT[degree,0.01745329251994328,AUTHORITY[EPSG,9122]],AUTHORITY[EPSG,4326]]

ORGANIZATION	ORGANIZATION_COORDSYS_ID
EPSG	4326

Système spatial de référence

- Permet de définir les paramètres des géométries :
  - Valeurs minimales et maximales de X et Y :
    - X (longitude) : -180 à +180
    - Y (latitude) : -90 à + 90
  - Unités de mesure
    - Mètres



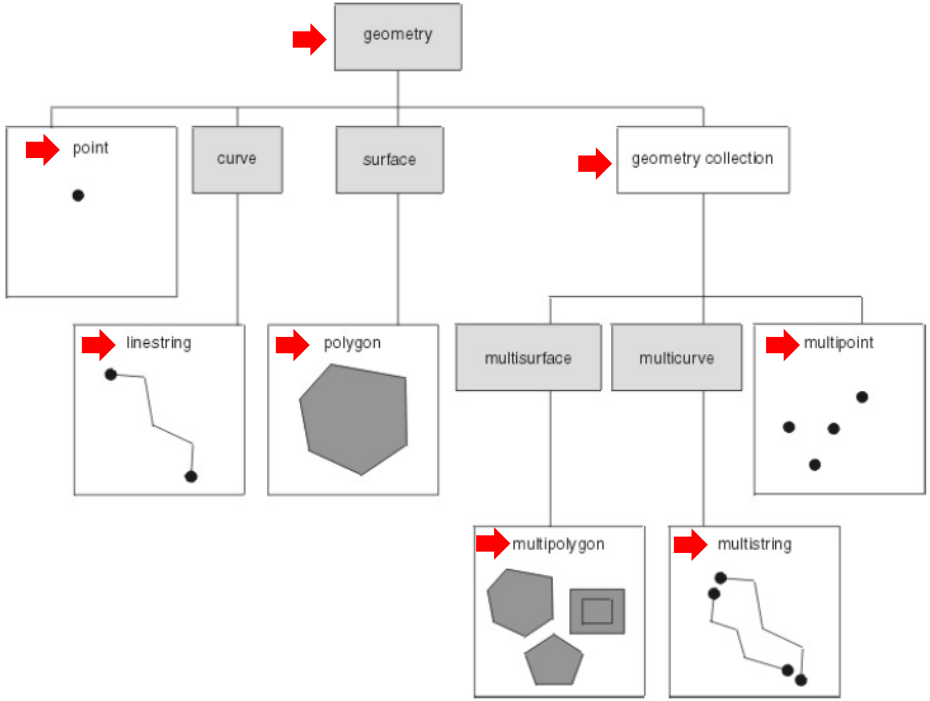
- Pour plus d'informations : visualiser le contenu de la vue **QSYS2.ST_SPATIAL_REFERENCE_SYSTEMS** du catalogue DB2

```
10 SELECT * FROM QSYS2.ST_SPATIAL_REFERENCE_SYSTEMS;
```

SPATIAL_REFERENCE_SYSTEM_NAME	SPATIAL_REFERENCE_SYSTEM_ID	MIN_X	MAX_X	MIN_Y	MAX_Y	COORDSYS_NAME
WGS84_SRS_4326	4326	-180.0	180.0	-90.0	90.0	GCS_WGS_1984

Données géospatiales dans DB2 for i

- 8 **nouveaux types de données** dans QSYS2 :
  - ST_Geometry
  - ST_Point
  - ST_LineString
  - ST_Polygon
  - ST_GeomCollection
  - ST_MultiPoint
  - ST_MultiLinestring
  - ST_MultiPolygon



- Sont de type BLOB

Données géospatiales dans DB2 for i

- **Point** (ST_Point) : emplacement identifié par longitude, latitude
  - Bâtiment, monument, magasin, ville…
- **Ligne** (ST_LineString) : ligne entre plusieurs points
  - Rue, route, canal, rivière…
- **Polygone** (ST_Polygon) : surface délimitée par un ensemble de points
  - Territoire, parcelle, parc, forêt, lac…
- **Points multiples** (ST_MultiPoint) : ensemble de points
  - Agences bancaires d'une ville, magasins de bricolage d'un quartier…

Données géospatiales dans DB2 for i

- **Lignes multiples** (ST_MultiLinestring) : ensemble de lignes
  - Réseau autoroutier, chemins de randonnée d'un parc…
- **Polygones multiple**s (ST_MultiPolygon) : ensemble de polygones
  - Les parcelles cadastrales d'un individu, les lacs d'une région…
- **Collection de géométries** (ST_GeomCollection) : ensemble de points et/ou de lignes et/ou de polygones
  - Rivières et lacs qui forment un bassin versant
- **Géométrie** (ST_Geometry) : type générique pouvant valoir un des types précédents

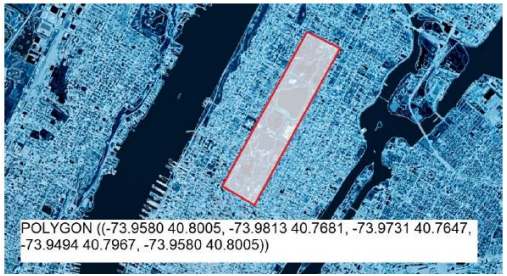
Données géospatiales dans DB2 for i

- Exemples de géométries :

ST_POINT
Statue de la liberté

ST_LINESTRING
Pont de Brooklyn

ST_POLYGON
Central Parc



Ajout d'une colonne de type "géospatial"

- Par **CREATE TABLE** ou **ALTER TABLE**

```
CREATE OR REPLACE TABLE cinemas
 (IDCINE INT AS IDENTITY IMPLICITLY HIDDEN PRIMARY KEY,
 NOM VARCHAR(30),
 ADRESSE VARCHAR(100),
 CODE_POSTAL FOR COLUMN CODPOS VARCHAR(5),
 VILLE VARCHAR(50),
 EMPLACEMENT FOR COLUMN EMPLACMT QSYS2.ST_POINT);
```

```
CREATE OR REPLACE TABLE territoires_commerciaux FOR SYSTEM NAME tercommerc
 (id_territoire     FOR COLUMN idterr     INT AS IDENTITY IMPLICITLY HIDDEN PRIMARY KEY,
  nom_territoire FOR COLUMN nomterr VARCHAR(30),
  zone_territoire FOR COLUMN zoneterr QSYS2.ST_POLYGON);
```

```
ALTER TABLE clients ADD COLUMN EMPLACEMENT FOR COLUMN EMPLACMT QSYS2.ST_POINT;
```

Fonctions géospatiales dans DB2 for i

- Il existe 51 fonctions (UDF ou UDTF) regroupées en 6 types :

Fonctions "Constructeur"
(pour insérer des données géospatiales)
- ST_Geometry
- ST_Point
- ST_LineString
- ST_Polygon
- ST_GeomCollection
- ST_MultiPoint
- ST_MultiLinestring
- ST_MultiPolygon
- ST_WKTToSQL
- ST_WKBToSQL

Fonctions de récupération des propriétés de géométries
- ST_Area
- ST_GeometryType
- ST_IsSimple
- ST_IsValid
- ST_MaxX
- ST_MaxY
- ST_MinX
- ST_MinY
- ST_NumPoints
- ST_SrsID
- ST_SrsName

Fonctions de géohachage
- ST_FuzzyGeohashCover
- ST_FuzzyGeohashCoverExtend
- ST_Geohash
- ST_GeohashCover
- ST_GeohashCoverExtend
- ST_Geohashvalue

Fonctions géospatiales dans DB2 for i

- Il existe 51 fonctions (UDF ou UDTF) regroupées en 6 types :

Fonctions de conversion de géométries
- ST_AsText
- ST_AsBinary
- ST_ToPoint
- ST_ToLineString
- ST_ToPolygon
- ST_ToMultiPoint
- ST_ToMultiLine
- ST_ToMultiPolygon

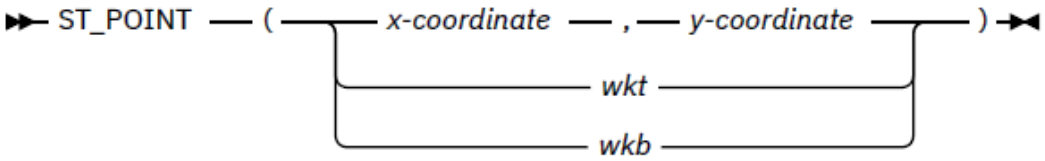
Fonctions de comparaison de géométries
- ST_Contains
- ST_Covers
- ST_Crosses
- ST_Difference
- ST_Disjoint
- ST_Distance
- ST_Equals
- ST_Intersects
- ST_Overlaps
- ST_Touches
- ST_Within

Fonctions de construction de nouvelles géométries
- ST_Buffer
- ST_Difference
- ST_Intersection
- ST_SymDifference
- ST_Union

Ajout de données géospatiales

- Il faut passer par les fonctions "constructeur"
  - Chaque type de données géospatial dispose de sa fonction constructeur
  - Exemple : la fonction **ST_POINT** permettra de transformer les coordonnées X et Y en format binaire

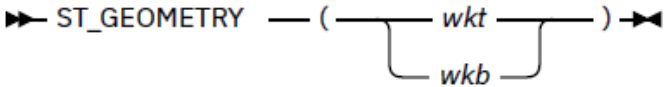
Fonctions "Constructeur"
- ST_Geometry
- ST_Point
- ST_LineString
- ST_Polygon
- ST_GeomCollection
- ST_MultiPoint
- ST_MultiLinestring
- ST_MultiPolygon
- ST_WKTToSQL
- ST_WKBToSQL

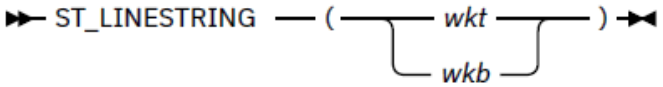


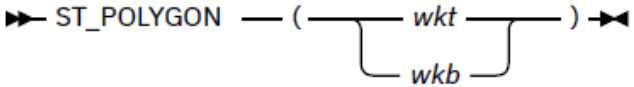
Cette fonction reçoit les coordonnées X et Y :
- Soit en format numérique
- Soit en format **WKT** (voir pages suivantes)
- Soit en format **WKB** (voir pages suivantes)

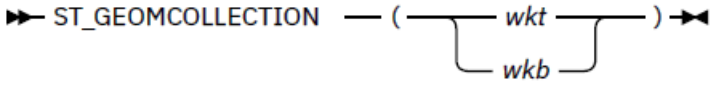
Ajout de données géospatiales

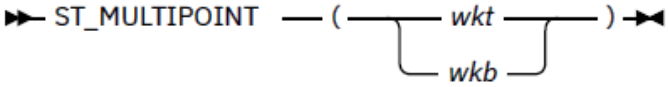
- Il faut passer par les fonctions "constructeur"
  - Chaque type de données géospatial a sa fonction constructeur
  - Pour les autres fonctions **autres que** ST_POINT, **seuls** les formats **WKT** et **WKB** sont supportés

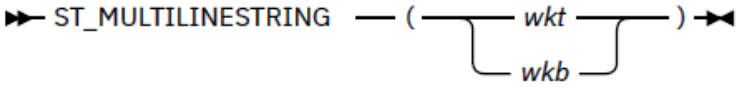


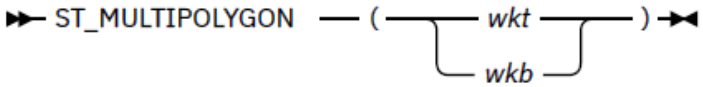










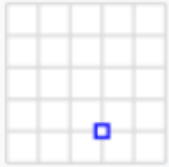
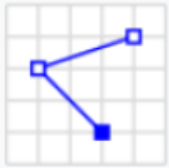


Ajout de données géospatiales

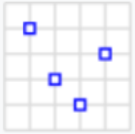
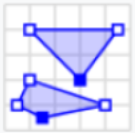
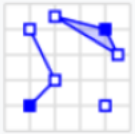
- Format **WKT** (Well Know Text) : format texte pour représenter des géométries
  - Définis par l'OGC (Open Geospatial Consortium)
  - Syntaxe :
    - `'POINT (x y)'`
    - `'LINESTRING (x1 y1, x2 y2, x3 y3)'`
    - `'POLYGON ((x1 y1, x2 y2, x3 y3, x1 y1))'`
    - `'GEOMETRYCOLLECTION(POINT(x y), LINESTRING(x1 y1, x2 y2))'`

- Il existe un équivalent binaire (**WKB** – Well Known Binary)
  - Utile pour transférer les données

Format WKT – Exemples

Point		`POINT (30 10)`
LineString		`LINESTRING (30 10, 10 30, 40 40)`
Polygon		`POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))`
		`POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))`

Format WKT – Exemples

MultiPoint		`MULTIPOINT ((10 40), (40 30), (20 20), (30 10))`
		`MULTIPOINT (10 40, 40 30, 20 20, 30 10)`
MultiLineString		`MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))`
MultiPolygon		`MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))`
		`MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)), ((20 35, 10 30, 10 10, 30 5, 45 20, 20 35), (30 20, 20 15, 20 25, 30 20)))`
GeometryCollection		`GEOMETRYCOLLECTION (POINT (40 10), LINESTRING (10 10, 20 20, 10 40), POLYGON ((40 40, 20 45, 45 30, 40 40)))`

Ajout de données géospatiales – Exemple

```
CREATE OR REPLACE TABLE cinemas
 (IDCINE INT AS IDENTITY IMPLICITLY HIDDEN PRIMARY KEY,
 NOM VARCHAR(30),
 ADRESSE VARCHAR(100),
 CODE_POSTAL FOR COLUMN CODPOS VARCHAR(5),
 VILLE VARCHAR(50),
 EMPLACEMENT FOR COLUMN EMPLACMT QSYS2.ST_POINT);
```

```
INSERT INTO cinemas VALUES
('Louis Jouvet',              '3 place Maurice Berteaux',  '78400',  'Chatou',              QSYS2.ST_POINT(2.156391, 48.886945)),
('Jean Marais',               '59 boulevard Carnot',       '78110',  'Le Vésinet',          QSYS2.ST_POINT('point(2.134326 48.893988)')),
('Ariel Centre-Ville',        '99 avenue Paul Doumer',     '92500',  'Rueil Malmaison',     QSYS2.ST_POINT('point(2.180852 48.879844)')),
('C2L',                       '25 rue du Vieux Marché',    '78100',  'Saint-Germain en Laye', QSYS2.ST_POINT('point(2.091527 48.896579)')),
('UGC Ciné Cité La Défense',  'Parvis de la Défense',      '92800',  'Puteaux',             QSYS2.ST_POINT('point(2.23494 48.89160)'));
```

Syntaxe : `QSYS2.ST_POINT('point(longitude latitude)')`

Pour récupérer les coordonnées : géocodage

- **Géocodage** : transformation d'adresses postales en coordonnées géographiques
  - Applications :
    - https://www.coordonnees-gps.fr/
    - https://www.google.fr/maps
    - …
  - Services Web :
    - https://adresse.data.gouv.fr/api-doc
    - https://developers.google.com/maps/documentation/geocoding?hl=fr
    - https://fr.open-street.com/doc/api/geocodage/
    - https://developer.precisely.com/apis/geocode
    - …

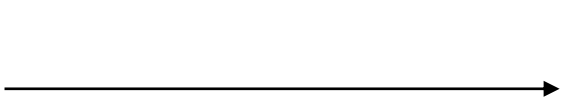
Géocodage : exemple

https://api-adresse.data.gouv.fr/search/?q=3+place+Maurice+Berteaux+78400+Chatou

JSON Données brutes En-têtes

Enregistrer Copier Tout réduire Tout développer Filtrer le JSON


```
type:                "FeatureCollection"
version:             "draft"
features:
  0:
    type:            "Feature"
    geometry:
      type:          "Point"
      coordinates:
        0:           2.156391
        1:           48.886945
    properties:
      label:         "3 Place Maurice Berteaux 78400 Chatou"
      score:         0.9674463636363636
      housenumber:   "3"
      id:            "78146_1230_00003"
      name:          "3 Place Maurice Berteaux"
      postcode:      "78400"
      citycode:      "78146"
      x:             638138.48
      y:             6865545
      city:          "Chatou"
      context:       "78, Yvelines, Île-de-France"
      type:          "housenumber"
      importance:    0.64191
      street:        "Place Maurice Berteaux"
  attribution:       "BAN"
  licence:           "ETALAB-2.0"
  query:             "3 place Maurice Berteaux 78400 Chatou"
  limit:             5
```


```
features:
  0:
    type:            "Feature"
    geometry:
      type:          "Point"
      coordinates:
        0:           2.156391
        1:           48.886945
```


Longitude

Latitude

Géocodage : exemple

- En **SQL** utilisation de 3 fonctions :
  - **HTTP_GET** pour appeler le Service Web
    - Syntaxe : **HTTP_GET(URL <, options (nom du magasin de certificats, proxy…)>)**
  - **ENCODE_URL** pour encoder l'adresse au format compatible URL
  - **JSON_TABLE** pour extraire du JSON reçu les coordonnées X et Y

```
54 VALUES QSYS2.HTTP_GET('https://api-adresse.data.gouv.fr/search/?q=3+place+Maurice+Berteaux+78400+Chatou');
55
56 VALUES QSYS2.HTTP_GET('https://api-adresse.data.gouv.fr/search/?q='
57                         CONCAT QSYS2.URL_ENCODE('3 place Maurice Berteaux 78400 Chatou'));
58
59 SELECT * FROM JSON_TABLE(QSYS2.HTTP_GET('https://api-adresse.data.gouv.fr/search/?q='
60                                          CONCAT QSYS2.URL_ENCODE('3 place Maurice Berteaux 78400 Chatou')),
61                                          '$.features[*].geometry' COLUMNS(x DOUBLE PATH '$.coordinates[0]',
62                                                                           y DOUBLE PATH '$.coordinates[1]' ));
```

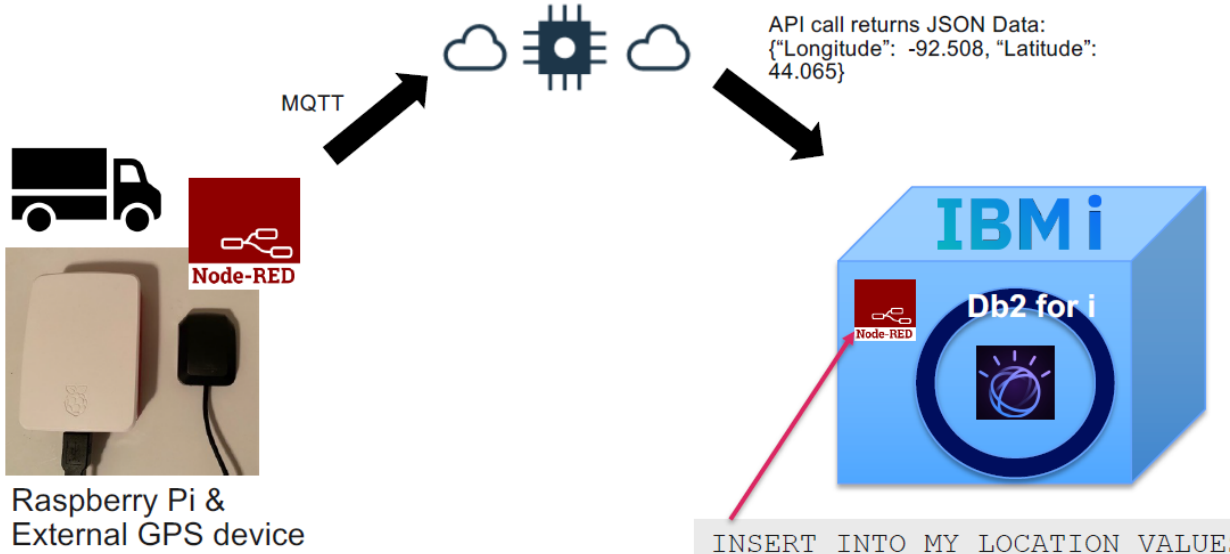
X	Y
2.156391	48.886945

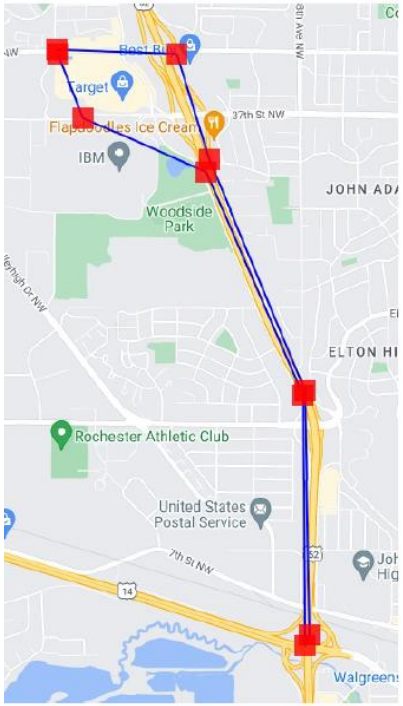
Ajout de données géospatiales – Exemple

```
INSERT INTO cinemas SELECT
'Louis Jouvet', '3 place Maurice Berteaux', '78400',  'Chatou', QSYS2.ST_POINT(x, y) FROM
JSON_TABLE(QSYS2.HTTP_GET('https://api-adresse.data.gouv.fr/search/?q='
                          CONCAT QSYS2.URL_ENCODE('3 place Maurice Berteaux 78400 Chatou')),
                          '$.features[*].geometry' COLUMNS(x DOUBLE PATH '$.coordinates[0]',
                                                           y DOUBLE PATH '$.coordinates[1]' ));
```

Ajout de données géospatiales

- Pour du "temps réel"



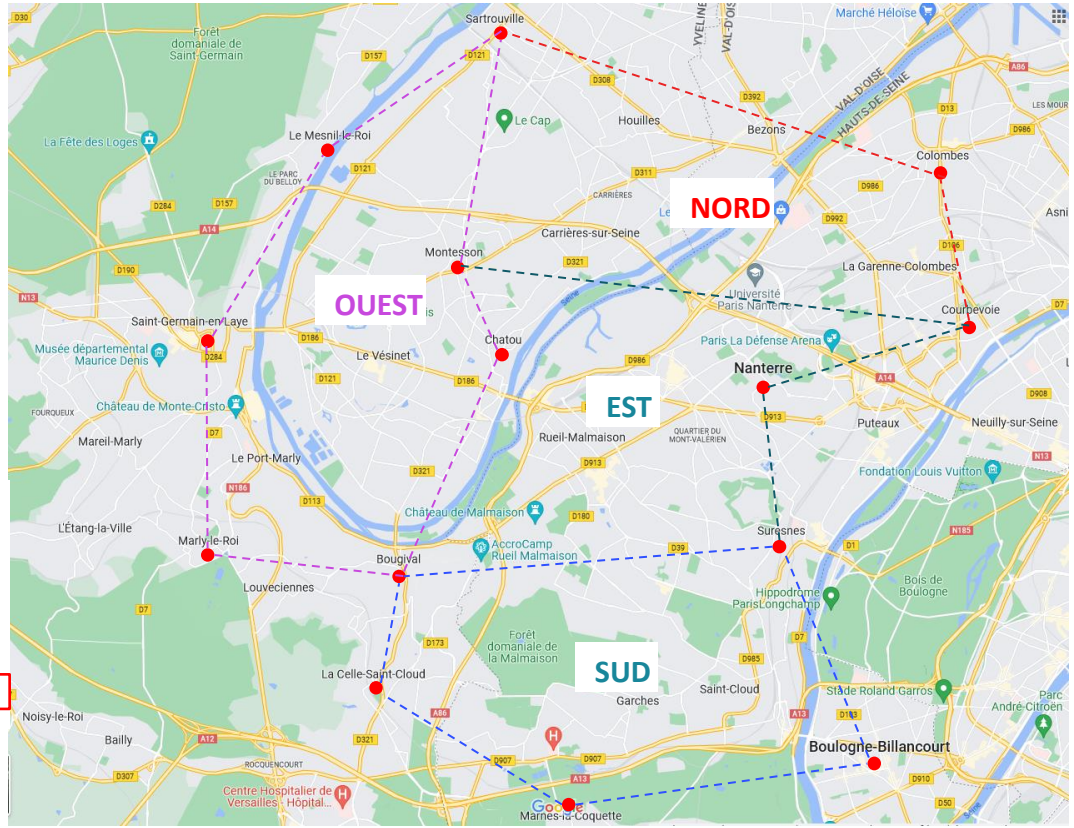



```
INSERT INTO MY_LOCATION VALUES
(QSYS2.ST_POINT(LONGITUDE, LATITUDE))
```

Ajout de données géospatiales – Exemple

- Table des **territoires commerciaux**
  - Le territoire commercial est défini par un **polygone**

```
CREATE OR REPLACE TABLE territoires_commerciaux
 FOR SYSTEM NAME tercommerc
 (id_territoire     FOR COLUMN idterr     INT
  AS IDENTITY IMPLICITLY HIDDEN PRIMARY KEY,
  nom_territoire FOR COLUMN nomterr VARCHAR(30),
  zone_territoire FOR COLUMN zoneterr QSYS2.ST_POLYGON);
```



Ajout de données géospatiales – Exemple

```
INSERT INTO territoires_commerciaux VALUES
('Territoire commercial OUEST',  QSYS2.ST_POLYGON('polygon((2.09698 48.89916, 2.09839 48.86860,
('Territoire commercial NORD',   QSYS2.ST_POLYGON('polygon((2.15670 48.94119, 2.15122 48.90891,
('Territoire commercial EST',    QSYS2.ST_POLYGON('polygon((2.15122 48.90891, 2.16059 48.89672,
('Territoire commercial SUD',    QSYS2.ST_POLYGON('polygon((2.13691 48.86569, 2.13638 48.84988,
. . .
. . .
                                  2.15122 48.90891, 2.15670 48.94119, 2.12295 48.92517, 2.09698 48.89916))')),
                                  2.15670 48.94119))')),
                                  2.21589 48.89265, 2.25777 48.90090, 2.15122 48.90891))')),
                                  2.22048 48.87011, 2.13691 48.86569))'));
```

Ajout de données géospatiales – Exemple

- Exemple de colonne de type **ST_GEOMETRY**

```
CREATE TABLE GEOMETRY_TABLE
  (GEO_ID INT, GEOMETRY QSYS2.ST_GEOMETRY);


INSERT INTO GEOMETRY_TABLE VALUES
  (1, QSYS2.ST_POINT('point(10 20)')),
  (2, QSYS2.ST_LINESTRING('linestring(10 20, 20 20)')),
  (3, QSYS2.ST_POLYGON('polygon((0 0, 0 10, 10 10, 10 0, 0 0))'));
```

Visualisation des données géospatiales

- Les données sont stockées en binaire. Pour les visualiser en clair il faut utiliser la fonction **ST_ASTEXT**

```
132 SELECT nom, adresse, code_postal, ville, emplacement, QSYS2.ST_ASTEXT(emplacement) AS coordonnees FROM cinemas;
```

NOM	ADRESSE	CODE_POSTAL	VILLE	EMPLACEMENT	COORDONNEES
Louis Jouvet	3 place Maurice Berteaux	78400	Chatou	01000000E6100000040...	POINT (2.1563909999999997 48.886945)
Jean Marais	59 boulevard Carnot	78110	Le Vésinet	01000000E6100000040...	POINT (2.1343259999999997 48.893988)
Ariel Centre-Ville	99 avenue Paul Doumer	92500	Rueil Malmaison	01000000E6100000040...	POINT (2.180852 48.879844)
C2L	25 rue du Vieux Marché	78100	Saint-Germain en Laye	01000000E6100000040...	POINT (2.0915269999999997 48.896578999999996)
UGC Ciné Cité La Défense	Parvis de la Défense	92800	Puteaux	01000000E6100000040...	POINT (2.23494 48.8916)

```
136 SELECT nom_territoire, QSYS2.ST_ASTEXT(zone_territoire) AS coordonnees FROM territoires_commerciaux;
```

NOM_TERRITOIRE	COORDONNEES
Territoire commercial OUEST	POLYGON ((2.09698 48.899159999999995, 2.0983899999999998 48.8686, 2.13691 48.86569, 2.16059 48.896719999999995, 2.15122 48.908
Territoire commercial NORD	POLYGON ((2.1567 48.94119, 2.15122 48.90891, 2.25777 48.9009, 2.25353 48.92238, 2.1567 48.94119))
Territoire commercial EST	POLYGON ((2.15122 48.90891, 2.16059 48.896719999999995, 2.13691 48.86569, 2.22048 48.87011, 2.21589 48.892649999999996, 2.25777
Territoire commercial SUD	POLYGON ((2.13691 48.86569, 2.13638 48.84988, 2.17066 48.830349999999996, 2.24204 48.839299999999994, 2.22048 48.87011, 2.13691

Récupération des propriétés des géométries

Fonctions de récupération des propriétés de géométries

- ST_Area → Aire du polygone en m^2 (renvoyé en DOUBLE)
- ST_GeometryType
- ST_IsSimple
- ST_IsValid
- ST_MaxX
- ST_MaxY
- ST_MinX
- ST_MinY
- ST_NumPoints
- ST_SrsID
- ST_SrsName

```
144 SELECT nom_territoire, DEC(QSYS2.ST_AREA(zone_territoire)/1000000, 10, 2)
145                        AS surface_du_territoire_en_km2
146  FROM territoires_commerciaux ORDER BY surface_du_territoire_en_km2 DESC;
```

NOM_TERRITOIRE	SURFACE_DU_TERRITOIRE_EN_KM2
Territoire commercial OUEST	24.91
Territoire commercial SUD	24.65
Territoire commercial EST	22.68
Territoire commercial NORD	22.32

Récupération des propriétés des géométries

Fonctions de récupération des propriétés de géométries

- ST_Area
- ST_GeometryType
- ST_IsSimple
- ST_IsValid
- ST_MaxX
- ST_MaxY
- ST_MinX
- ST_MinY
- ST_NumPoints
- ST_SrsID
- ST_SrsName

```
142 SELECT QSYS2.ST_GEOMETRYTYPE(emplacement) FROM cinemas LIMIT 1;
```

00001
ST_POINT

```
143 SELECT QSYS2.ST_GEOMETRYTYPE(zone_territoire) FROM territoires_commerciaux LIMIT 1;
```

00001
ST_POLYGON

```
144 SELECT QSYS2.ST_GEOMETRYTYPE(geometry) FROM geometry_table;
```

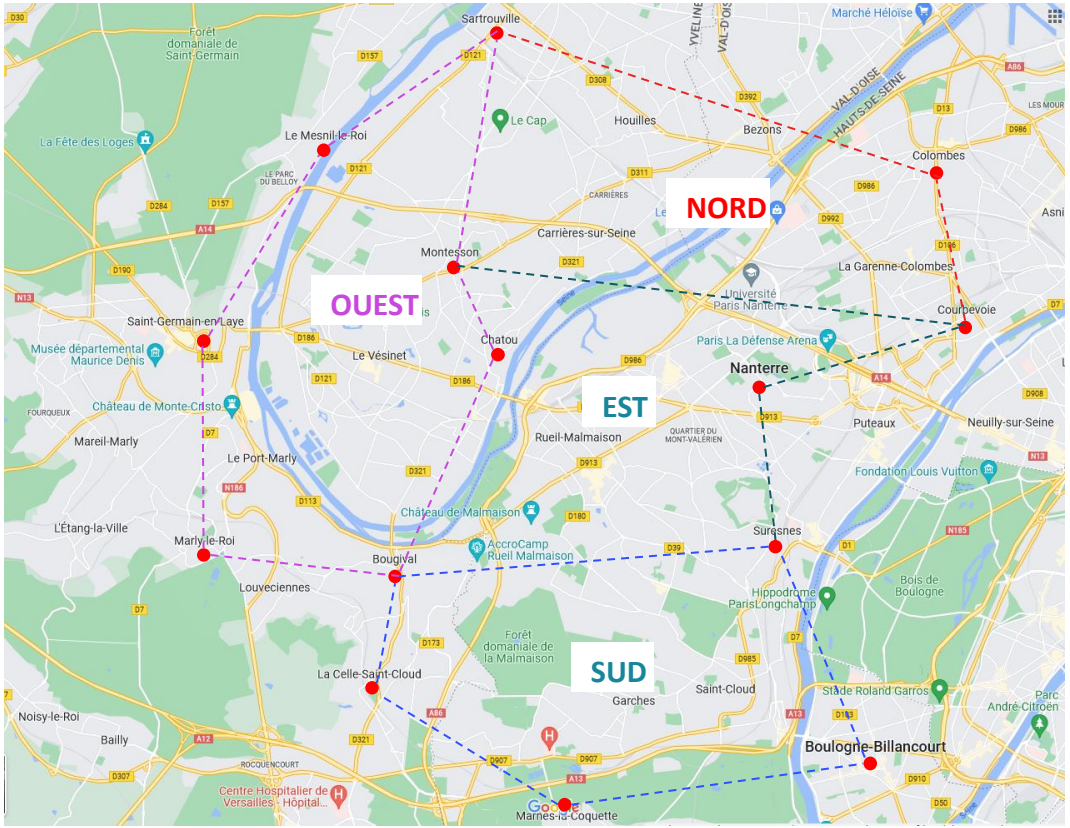
00001
ST_POINT
ST_LINESTRING
ST_POLYGON

Récupération des propriétés des géométries

Fonctions de récupération des propriétés de géométries

- ST_Area
- ST_GeometryType
- ST_IsSimple
- ST_IsValid
- ST_MaxX
- ST_MaxY
- ST_MinX
- ST_MinY
- ST_NumPoints
- ST_SrsID
- ST_SrsName

Quels sont les territoires les plus à l'ouest, est, sud, nord ?



Récupération des propriétés des géométries

```
-- Quel est(quels sont) le(s) territoire(s) qui s'étend(ent) le plus à l'ouest ?
SELECT nom_territoire FROM territoires_commerciaux WHERE QSYS2.ST_MINX(zone_territoire) =
                                                        (SELECT MIN(QSYS2.ST_MINX(zone_territoire))
                                                         FROM territoires_commerciaux);
```

NOM_TERRITOIRE
Territoire commercial OUEST

```
-- Quel est(quels sont) le(s) territoire(s) qui s'étend(ent) le plus à l'est ?
SELECT nom_territoire FROM territoires_commerciaux WHERE QSYS2.ST_MAXX(zone_territoire) =
                                                        (SELECT MAX(QSYS2.ST_MAXX(zone_territoire))
                                                         FROM territoires_commerciaux);
```

NOM_TERRITOIRE
Territoire commercial NORD
Territoire commercial EST

```
-- Quel est(quels sont) le(s) territoire(s) qui s'étend(ent) le plus au sud ?
SELECT nom_territoire FROM territoires_commerciaux WHERE QSYS2.ST_MINY(zone_territoire) =
                                                        (SELECT MIN(QSYS2.ST_MINY(zone_territoire))
                                                         FROM territoires_commerciaux);
```

NOM_TERRITOIRE
Territoire commercial SUD

```
-- Quel est(quels sont) le(s) territoire(s) qui s'étend(ent) le plus au nord ?
SELECT nom_territoire FROM territoires_commerciaux WHERE QSYS2.ST_MAXY(zone_territoire) =
                                                        (SELECT MAX(QSYS2.ST_MAXY(zone_territoire))
                                                         FROM territoires_commerciaux);
```

NOM_TERRITOIRE
Territoire commercial OUEST
Territoire commercial NORD

Comparaison des géométries – DISTANCE

Fonctions de comparaison de géométries

- ST_Contains
- ST_Covers
- ST_Crosses
- ST_Difference
- ST_Disjoint
- ST_Distance
- ST_Equals
- ST_Intersects
- ST_Overlaps
- ST_Touches
- ST_Within

Quel sont les cinémas le plus proche et le plus lointain de mon emplacement actuel ?

Comparaison des géométries – DISTANCE

```
-- Alimentation d'une variable avec les coordonnées de mon emplacement actuel
CREATE VARIABLE mon_emplacement_actuel QSYS2.ST_POINT;
SET mon_emplacement_actuel = QSYS2.ST_POINT('point(2.15835 48.90342)');
```

```
209 -- Quel est le cinéma le plus proche de mon emplacement actuel ?
210 SELECT nom, adresse, code_postal, ville,
211        ROUND((QSYS2.ST_DISTANCE(mon_emplacement_actuel, emplacement)/1000), 1) AS "Distance en km"
212  FROM cinemas ORDER BY "Distance en km" LIMIT 1;
```

NOM	ADRESSE	CODE_POSTAL	VILLE	Distance en km
Louis Jouvet	3 place Maurice Berteaux	78400	Chatou	1.8

```
214  -- Quel est le cinéma le plus loin de mon emplacement actuel ?
215 SELECT nom, adresse, code_postal, ville,
216        ROUND((QSYS2.ST_DISTANCE(mon_emplacement_actuel, emplacement)/1000), 1) AS "Distance en km"
217  FROM cinemas ORDER BY "Distance en km" DESC LIMIT 1;
```

NOM	ADRESSE	CODE_POSTAL	VILLE	Distance en km
UGC Ciné Cité La Défense	Parvis de la Défense	92800	Puteaux	5.8

Comparaison des géométries

Syntaxe : **ST_XXX**(géométrie_1 géométrie_2)

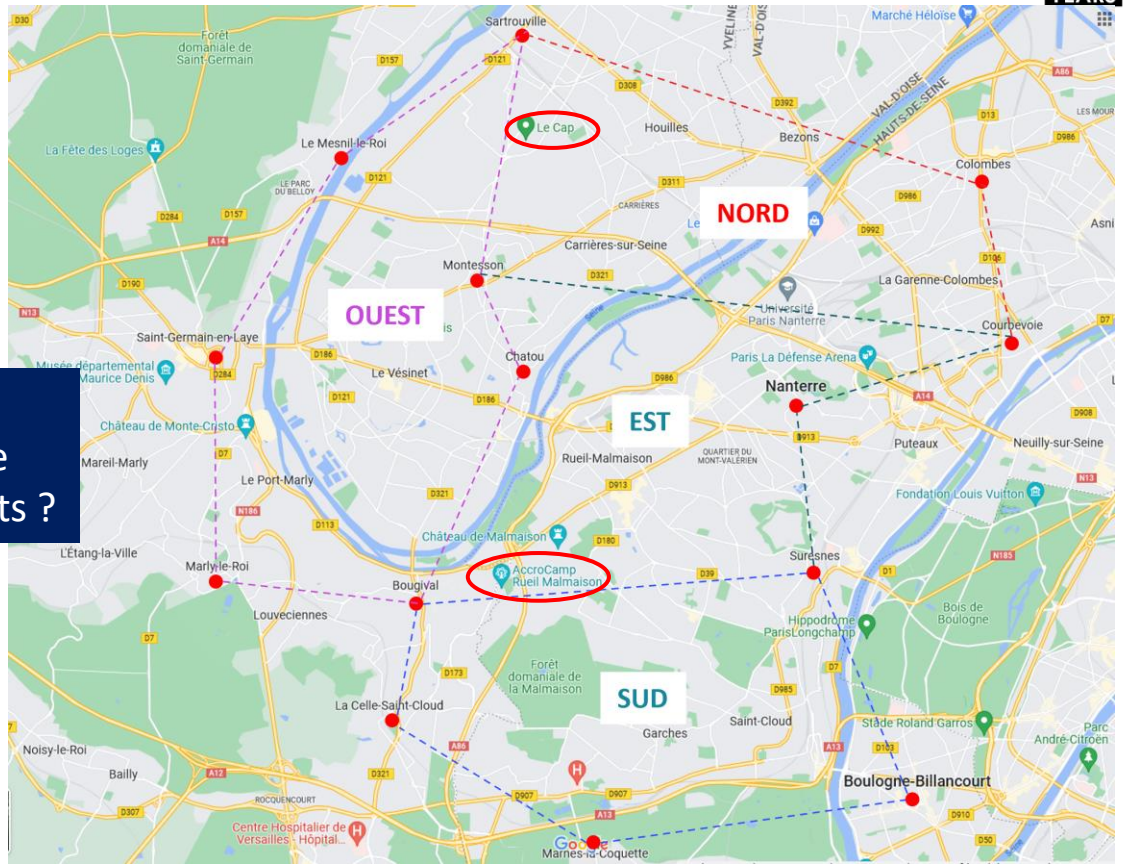
- **ST_EQUALS** Renvoie 1 si les 2 géométries sont strictement identiques
- **ST_INTERSECTS** Renvoie 1 si les 2 géométries ont une surface commune
- **ST_DISJOINT** Renvoie 1 si les 2 géométries n'ont aucune surface commune (opposé de ST_INTERSECTS)
- **ST_CROSSES** Renvoie 1 si les 2 géométries ont des points/surfaces en commun mais pas tout
- **ST_OVERLAPS** Renvoie 1 si les deux géométries se chevauchent ou se croisent mais que l'une ne contient pas complètement l'autre
- **ST_TOUCHES** Renvoie 1 si une géométrie touche l'autre mais sans intersection
- **ST_DIFFERENCE** Renvoie la géométrie commune aux 2 géométries
- **ST_WITHIN** Renvoie 1 si géométrie_1 est complètement à l'intérieur de géométrie_2
- **ST_CONTAINS** Renvoie 1 si géométrie_2 est complètement à l'intérieur de géométrie_1 (opposé de ST_WITHIN)
- **ST_COVERS** Renvoie 1 si géométrie_1 couvre complètement géométrie_2

Comparaison des géométries – CONTAINS

Fonctions de comparaison de géométries

- ST_Contains
- ST_Covers
- ST_Crosses
- ST_Difference
- ST_Disjoint
- ST_Distance
- ST_Equals
- ST_Intersects
- ST_Overlaps
- ST_Touches
- ST_Within

Quel territoire commercial couvre chacun de ces 2 clients ?



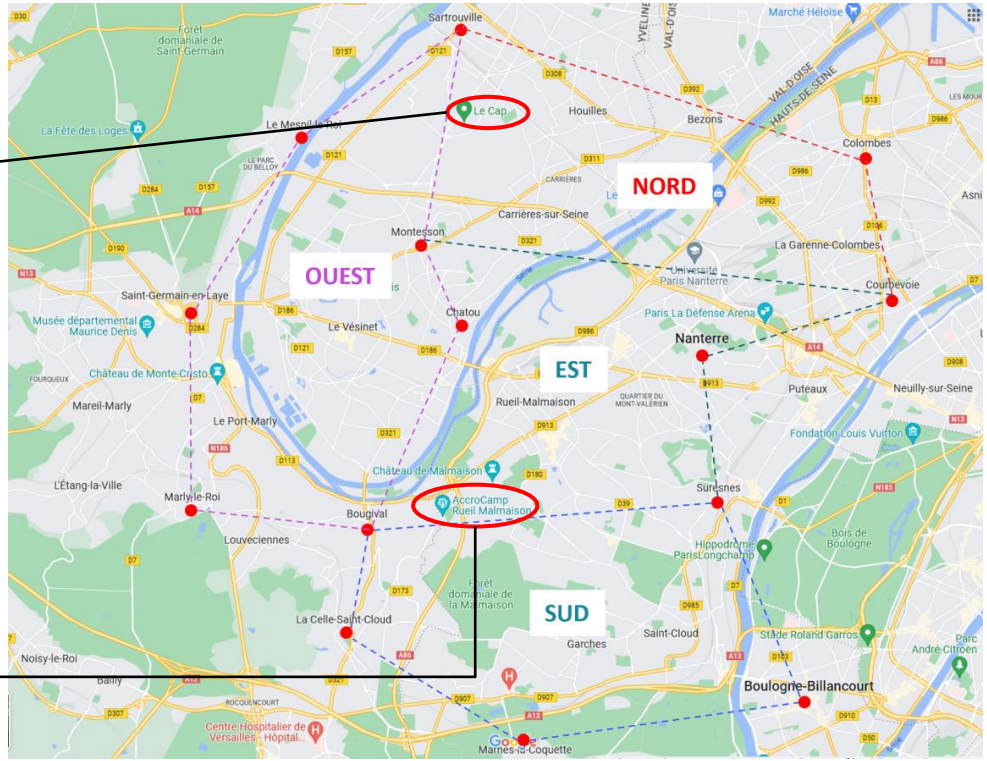
Comparaison des géométries – CONTAINS

```
220 SELECT nom_territoire FROM territoires_commerciaux
221  WHERE QSYS2.ST_CONTAINS(zone_territoire,
222         QSYS2.ST_POINT('point(2.16447 48.92703)')) = 1;
```

NOM_TERRITOIRE
Territoire commercial NORD

```
224 SELECT nom_territoire FROM territoires_commerciaux
225  WHERE QSYS2.ST_CONTAINS(zone_territoire,
226         QSYS2.ST_POINT('point(2.15511 48.86802)')) = 1;
```

NOM_TERRITOIRE
Territoire commercial EST

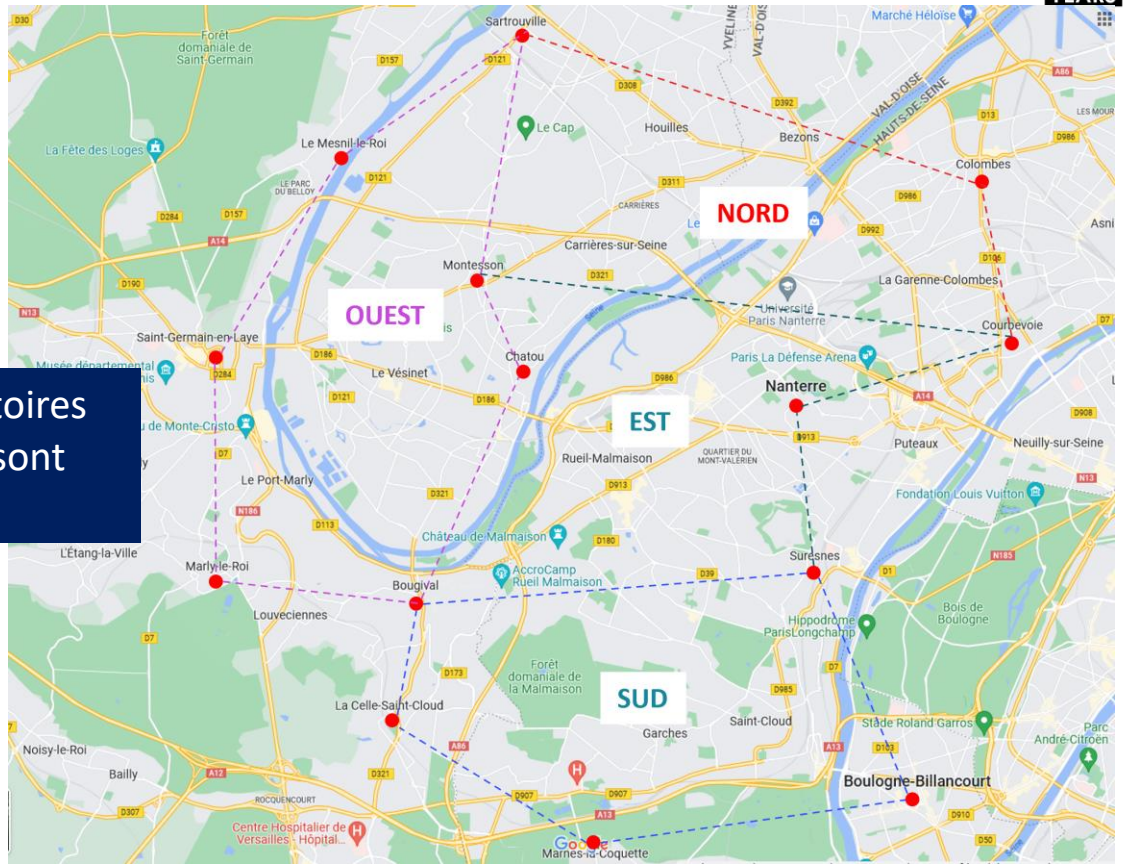


Comparaison des géométries – TOUCHES

Fonctions de comparaison de géométries

- ST_Contains
- ST_Covers
- ST_Crosses
- ST_Difference
- ST_Disjoint
- ST_Distance
- ST_Equals
- ST_Intersects
- ST_Overlaps
- ST_Touches
- ST_Within

Quels sont les territoires commerciaux qui sont contigus ?



Comparaison des géométries – TOUCHES

```
242 -- Quels sont les territoires commerciaux qui sont contigus ?
243 WITH cte AS (SELECT t1.nom_territoire AS t1nt, t2.nom_territoire AS t2nt FROM territoires_commerciaux t1
244                 CROSS JOIN territoires_commerciaux t2
245                 WHERE QSYS2.ST_TOUCHES(t1.zone_territoire, t2.zone_territoire) = 1)
246  SELECT DISTINCT CASE WHEN t1nt > t2nt THEN t1nt ELSE t2nt END AS territoire1,
247                       CASE WHEN t1nt > t2nt THEN t2nt ELSE t1nt END AS territoire2
248  FROM cte;
```

TERRITOIRE1	TERRITOIRE2
Territoire commercial OUEST	Territoire commercial NORD
Territoire commercial OUEST	Territoire commercial EST
Territoire commercial NORD	Territoire commercial EST
Territoire commercial SUD	Territoire commercial EST

Fonctions de création de géométries

Fonctions de construction de nouvelles géométries
- ST_Buffer
- ST_Difference
- ST_Intersection
- ST_SymDifference
- ST_Union

Syntaxe : **ST_BUFFER**(géométrie_1 x)

- **ST_BUFFER** Crée une géométrie à partir de géométrie_1 "augmentée" de x mètres :
  - Si géométrie_1 est un **point** : un "cercle" de x mètres de rayon autour du point
  - Si géométrie_1 est une **ligne** : un polygone de x mètres autour de la ligne
  - Si géométrie_1 est un **polygone** : un polygone de x mètres autour du polygone

Syntaxe : **ST_XXX**(géométrie_1 géométrie_2)

- **ST_UNION** L'union de géométrie_1 et géométrie_2
- **ST_INTERSECTION** L'intersection de géométrie_1 et géométrie_2
- **ST_DIFFERENCE** La partie de géométrie_1 qui n'appartient pas à géométrie_2
- **ST_SYMDIFFERENCE** La partie de géométrie_1 qui n'appartient pas à géométrie_2 + la partie de géométrie_2 qui n'appartient pas à géométrie_1

Fonctions de création de géométries – BUFFER

- Quels sont mes concurrents dans un rayon de 10 km ?
- Existe-t-il des agences bancaires dans un rayon de 500 m ?
- Jusqu'où s'étend la zone de sécurité de l'usine de produits chimiques ?
- Quel est l'impact sur les habitations existantes si on élargit la rue de 1,5m de chaque côté pour créer une piste cyclable ?

Fonctions de création de géométries – BUFFER

- Un **BUFFER** à partir d'un **point** renvoie un **"cercle"** de x mètres de **rayon** autour du point :
  - En fait c'est un **octogone** qui est renvoyé (le cercle est inscrit **dans** l'octogone)

```
262 VALUES QSYS2.ST_ASTEXT(QSYS2.ST_BUFFER(QSYS2.ST_POINT('point(2.15835 48.90342)'), 100));
```

```
00001
POLYGON ((2.157784 48.904317999999996, 2.156983 48.903791999999996, 2.156983 48.903048, 2.157784 48.902522,
```

```
2.158916 48.902522, 2.159717 48.903048, 2.159717 48.903791999999996, 2.158916 48.904317999999996, 2.157784 48.904317999999996))
```

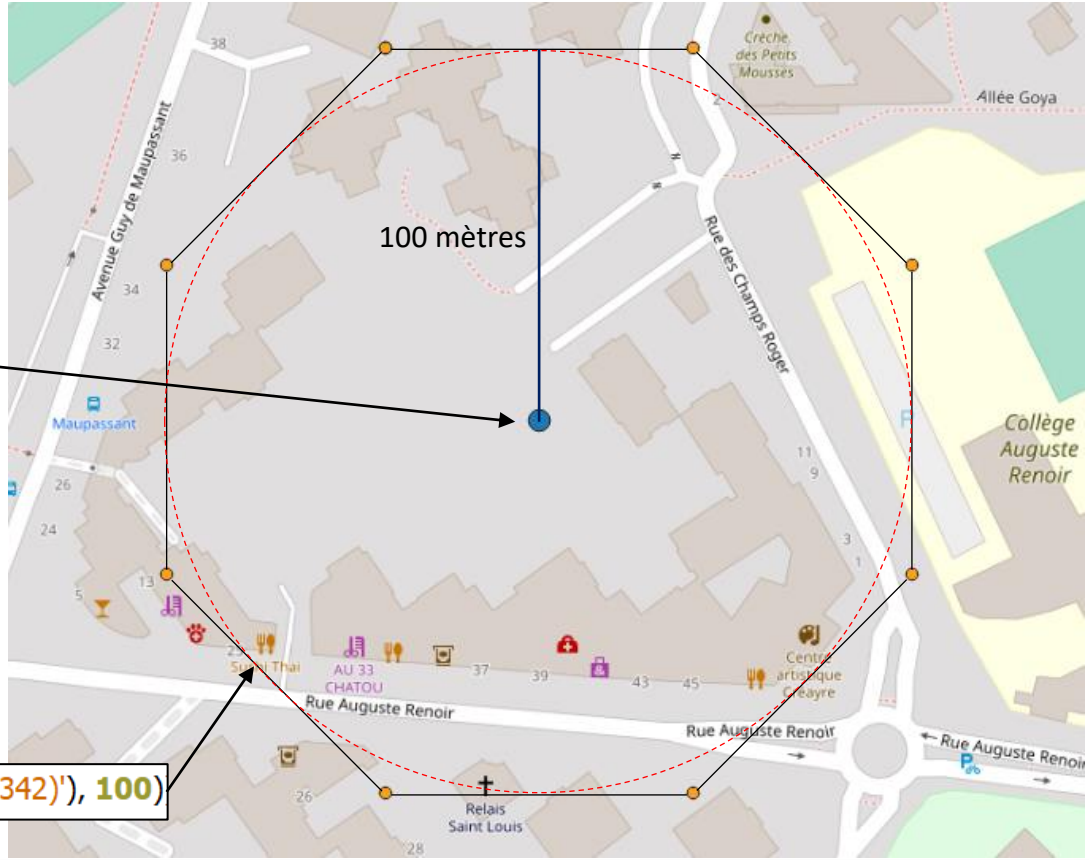
Fonctions de création de géométries – BUFFER

- BUFFER à partir d'un **point**

```
264 VALUES QSYS2.ST_AREA(
265            QSYS2.ST_BUFFER(
266              QSYS2.ST_POINT(
267               'point(2.15835 48.90342)'), 100))
268               /10000;
```

```
00001
 3.3137067174250197
```

Aire en km² (aire de l'octogone)




```
QSYS2.ST_BUFFER(QSYS2.ST_POINT('point(2.15835 48.90342)'), 100)
```

Fonctions de création de géométries – BUFFER

- BUFFER à partir d'un **point**

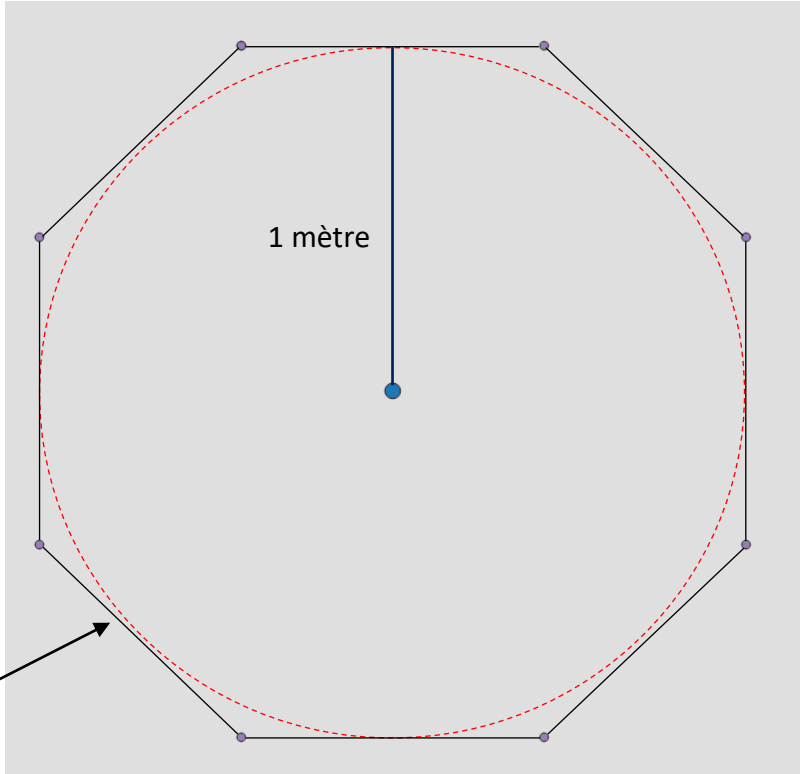
```
281 VALUES QSYS2.ST_AREA(
282            QSYS2.ST_BUFFER(
283              QSYS2.ST_POINT(
284               'point(2.15835 48.90342)'), 1));
```

```
00001
 5.492012179961215
```

BUG

Aire en m^2 (aire de l'octogone)

```
QSYS2.ST_BUFFER(QSYS2.ST_POINT('point(2.15835 48.90342)'), 1)
```



Fonctions de création de géométries – BUFFER

- BUFFER à partir d'une **ligne**

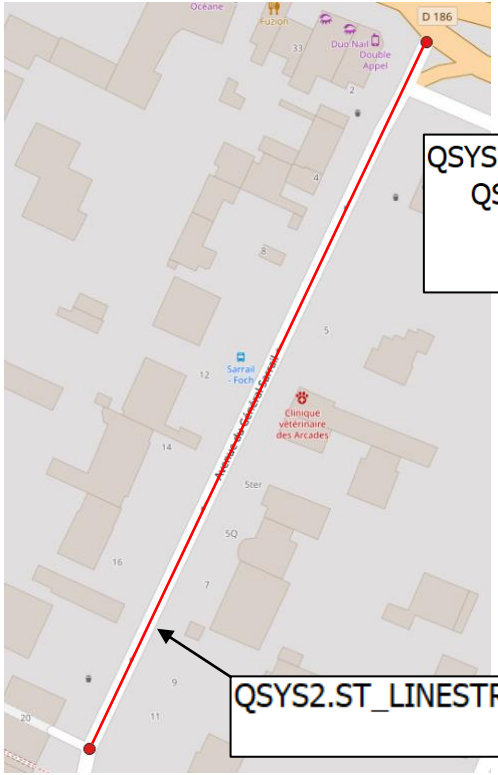
```
VALUES  QSYS2.ST_ASTEXT(QSYS2.ST_BUFFER(
          QSYS2.ST_LINESTRING('linestring(2.15637321 48.88976598,
                                          2.15539963 48.88841924)'),
                              3));
```

renvoie un polygone :

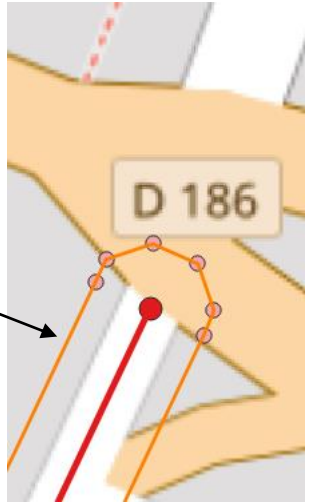
```
POLYGON ((2.155437 48.888407, 2.1564099999999997 48.889753999999996,
2.156417 48.889765, 2.156406 48.889786, 2.1563749999999997 48.889795,
2.156343 48.889787999999996, 2.156336 48.889778, 2.155363 48.888431,
2.155356 48.888419999999996, 2.155367 48.888399, 2.155398 48.88839,
2.15543 48.888397, 2.155437 48.888407))
```

Fonctions de création de géométries – BUFFER

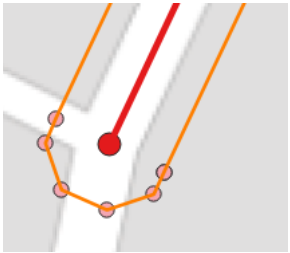
- BUFFER à partir d'une **ligne**




```
QSYS2.ST_BUFFER(
    QSYS2.ST_LINESTRING('linestring(2.15637321 48.88976598,
                                    2.15539963 48.88841924)'),
                        3));
```



. . .



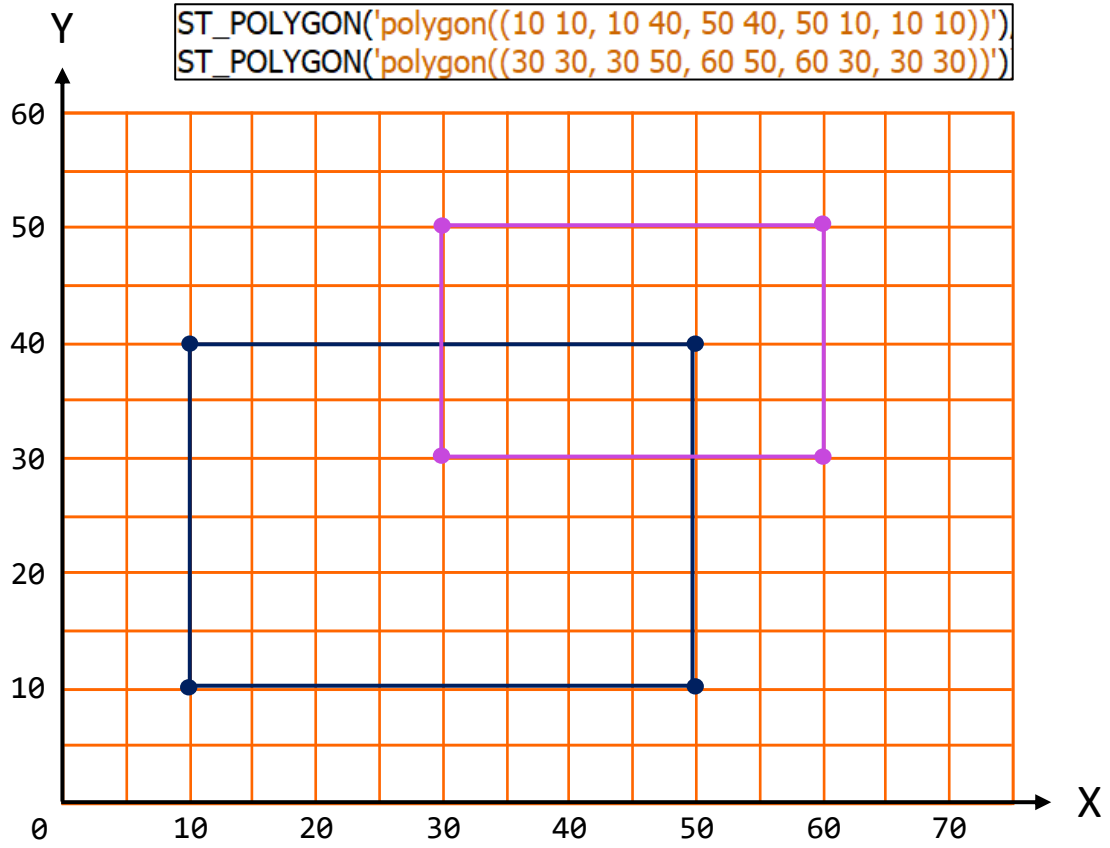
```
QSYS2.ST_LINESTRING('linestring(2.15637321 48.88976598,
                                2.15539963 48.88841924)')
```

Fonctions de création de géométries

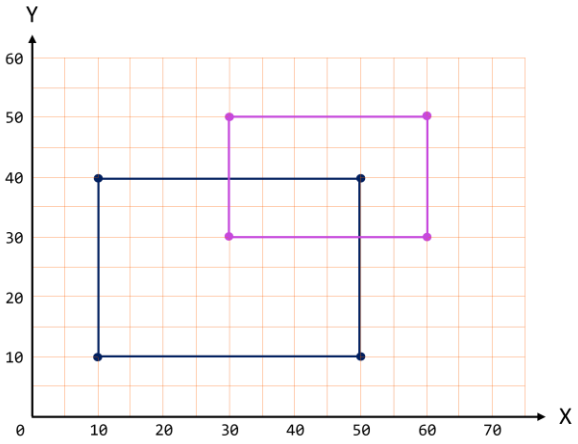
Fonctions de construction de nouvelles géométries

- ST_Buffer
- ST_Difference
- ST_Intersection
- ST_SymDifference
- ST_Union

Situation d'origine

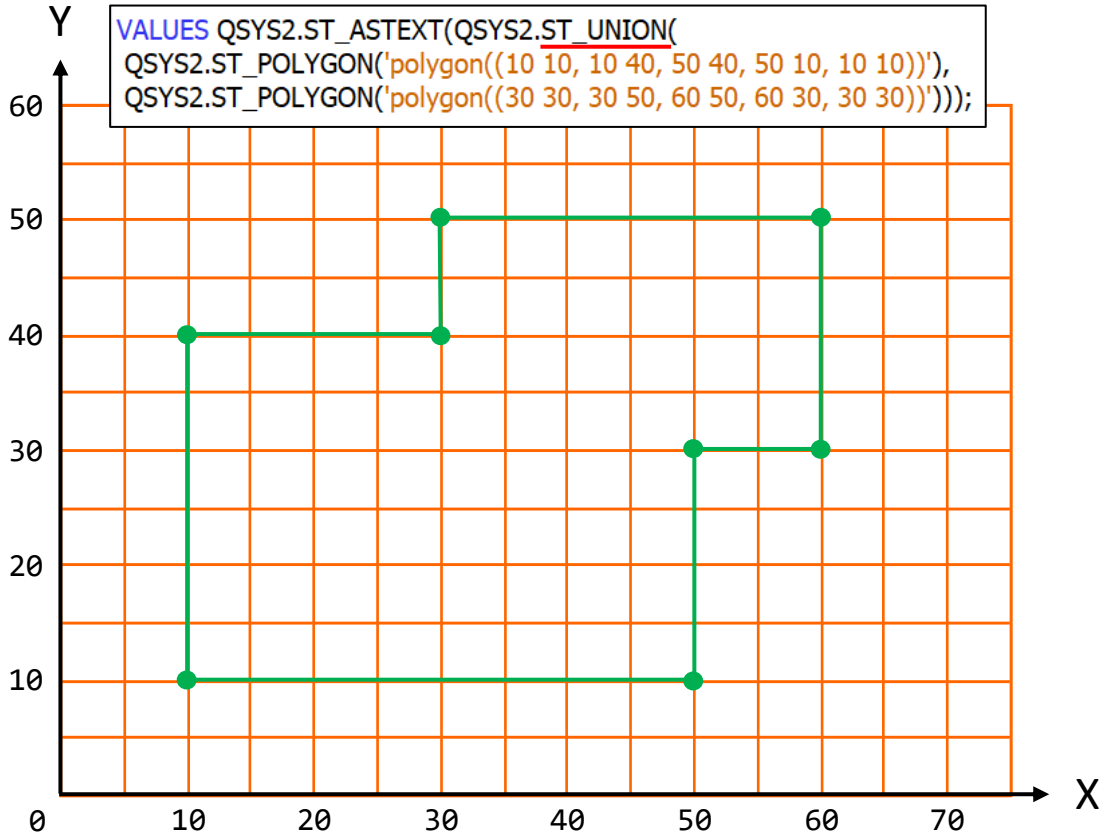


Fonctions de création de géométries – UNION

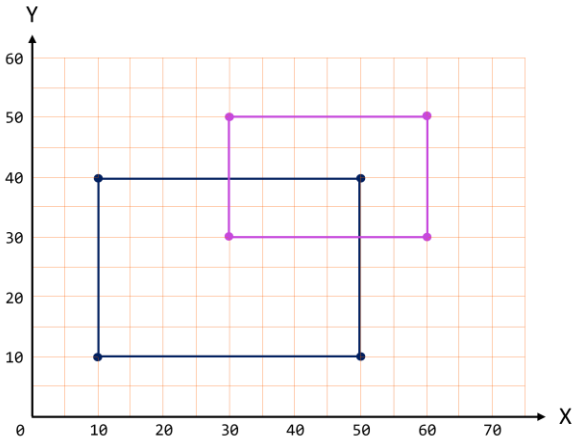



```
POLYGON ((60 30, 60 50,
30 50, 30 40, 10 40,
10 10, 50 10, 50 30,
60 30))
```

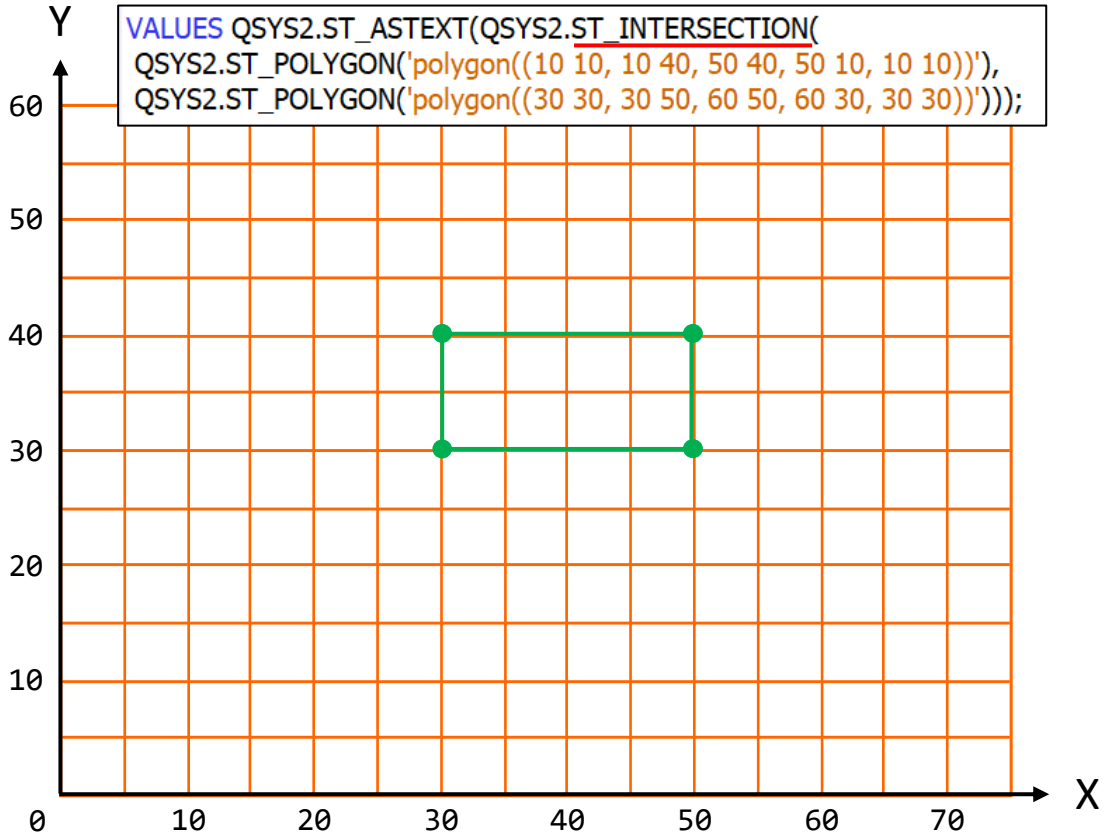
```
VALUES QSYS2.ST_ASTEXT(QSYS2.ST_UNION(
 QSYS2.ST_POLYGON('polygon((10 10, 10 40, 50 40, 50 10, 10 10))'),
 QSYS2.ST_POLYGON('polygon((30 30, 30 50, 60 50, 60 30, 30 30))')));
```



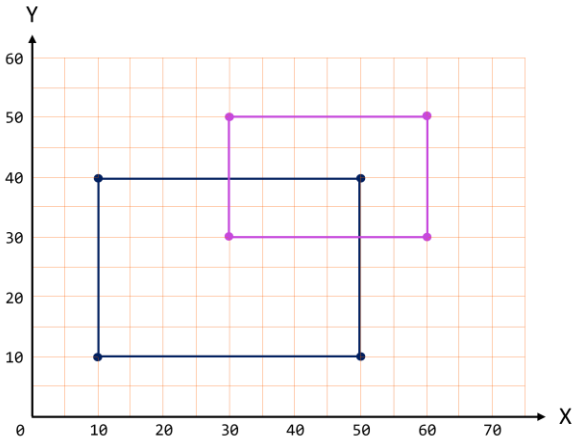
Fonctions de création de géométries – INTERSECTION




```
POLYGON ((30 30, 30 40,
50 40, 50 30, 30 30))
```

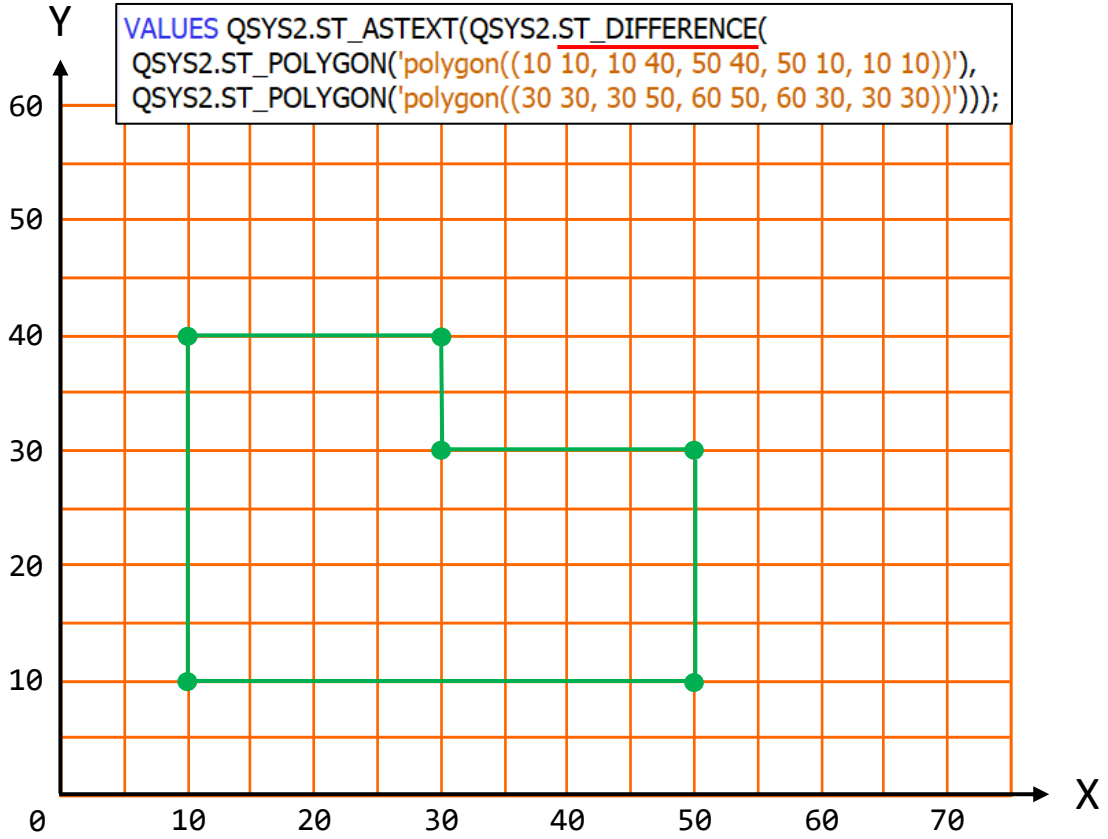


Fonctions de création de géométries – DIFFERENCE

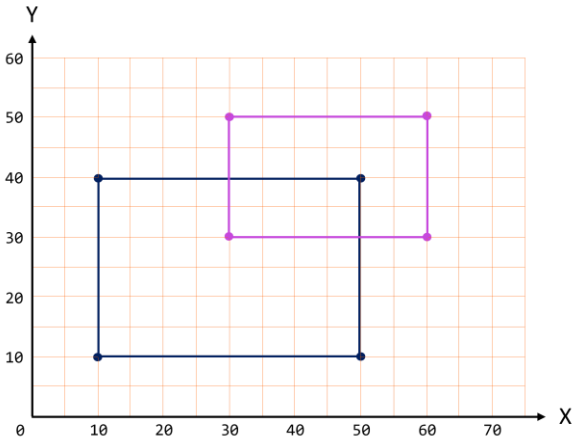



```
POLYGON ((10 10, 10 40,
30 40, 30 30, 50 30,
50 10, 10 10))
```

```
VALUES QSYS2.ST_ASTEXT(QSYS2.ST_DIFFERENCE(
 QSYS2.ST_POLYGON('polygon((10 10, 10 40, 50 40, 50 10, 10 10))'),
 QSYS2.ST_POLYGON('polygon((30 30, 30 50, 60 50, 60 30, 30 30))')));
```

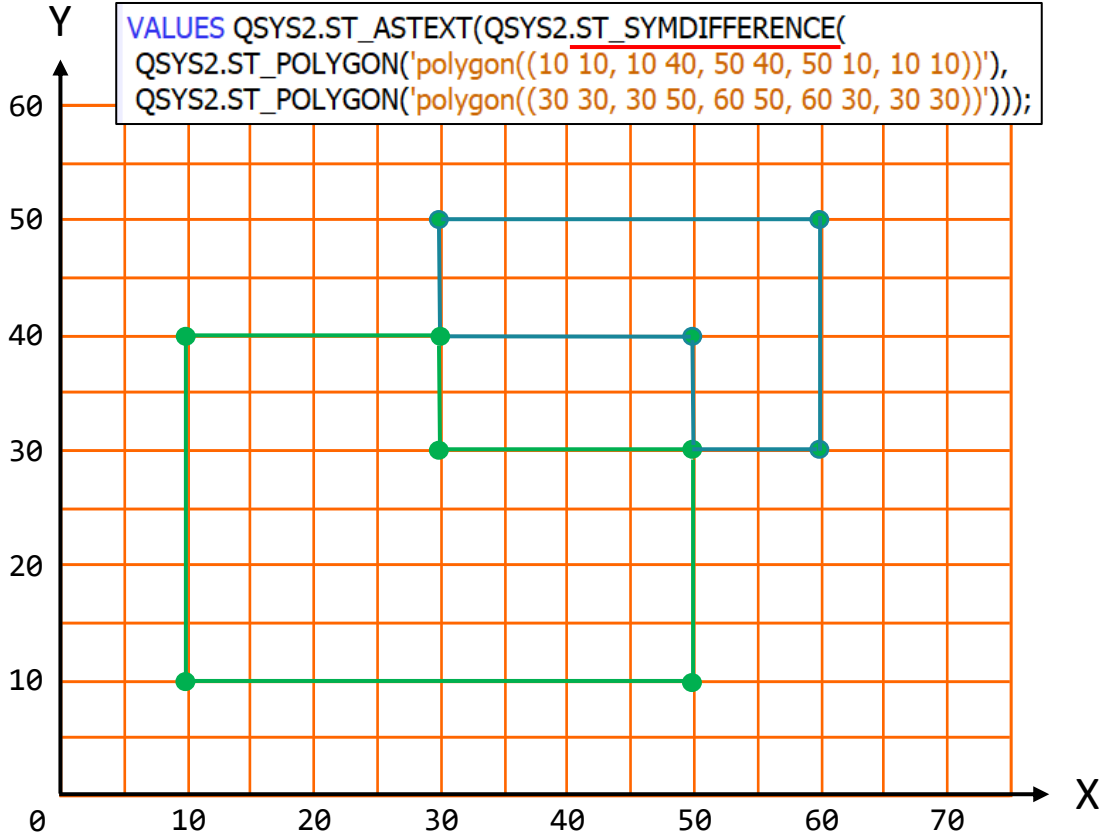


Fonctions de création de géométries – SYMDIFFERENCE




```
MULTIPOLYGON (((50 40, 50 30,
60 30, 60 50, 30 50, 30 40,
50 40)),
((50 30, 30 30, 30 40, 10 40,
10 10, 50 10, 50 30)))
```

```
VALUES QSYS2.ST_ASTEXT(QSYS2.ST_SYMDIFFERENCE(
 QSYS2.ST_POLYGON('polygon((10 10, 10 40, 50 40, 50 10, 10 10))'),
 QSYS2.ST_POLYGON('polygon((30 30, 30 50, 60 50, 60 30, 30 30))')));
```



Fonctions de création de géométries – UNION

```
347 -- Quels sont les clients qui sont couverts par le territoire commercial NORD-OUEST
348 SELECT idclient, societe, ville FROM clients
349  WHERE QSYS2.ST_CONTAINS(
350            QSYS2.ST_UNION((SELECT zone_territoire FROM territoires_commerciaux
351                              WHERE nom_territoire LIKE '%NORD'),
352                           (SELECT zone_territoire FROM territoires_commerciaux
353                             WHERE nom_territoire LIKE '%OUEST')),
354                          emplacement) = 1;
```

IDCLIENT	SOCIETE	VILLE
6	Le Cap	Sartrouville
10	Paintball 75	Bougival

Table **CLIENTS**

Nom de colonne	Nom de système	Type de données
IDCLIENT	IDCLIENT	INTEGER
SOCIETE	SOCIETE	VARCHAR
ADRESSE	ADRESSE	VARCHAR
CODE_POSTAL	CODPOS	VARCHAR
VILLE	VILLE	VARCHAR
EMPLACEMENT	EMPLACMT	QSYS2.ST_POINT

Table **TERRITOIRES_COMMERCIAUX**

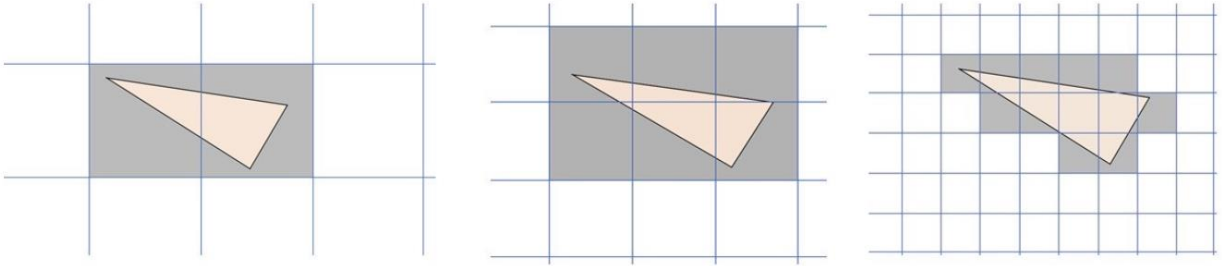
Nom de colonne	Nom de système	Type de données
ID_TERRITOIRE	IDTERR	INTEGER
NOM_TERRITOIRE	NOMTERR	VARCHAR
ZONE_TERRITOIRE	ZONETERR	QSYS2.ST_POLYGON

Géohachage – Geohash

- **GEOHASH** est un **système de géocodage des coordonnées géographiques** (latitude / longitude), dans le domaine public. Le système est basé sur une fonction de hachage qui subdivise la surface terrestre selon une grille hiérarchique

Géohachage – Geohash

- Un **geohash** est un **nombre** qui identifie de manière **unique** une région spécifique. L'algorithme geohash divise la Terre en régions, appelées **cellules**, et convertit la latitude et la longitude du centre de chaque cellule en un nombre qui l'identifie de manière unique. La taille de chaque cellule est déterminée par la valeur de **profondeur**, qui est spécifiée par l'utilisateur de l'algorithme (valeur entière comprise entre 1 et 45). *Plus la valeur de la profondeur est faible, plus la taille de la cellule est grande*



Géohachage – Geohash

Profondeur	Taille approximative de la cellule	Dimensions approximatives	Exemples
45	23 m^2	4,8 m * 4,8 m	Localisation GPS, bâtiment
28	3 km^2	2,45 km * 1,25 km	Terrain, parcelle
23	100 km^2	9,8 km * 9,8 km	Forêt
18	3000 km^2	78,3 km * 19,1 km	Département
13	100.000 km^2	315 km * 315 km	Pays

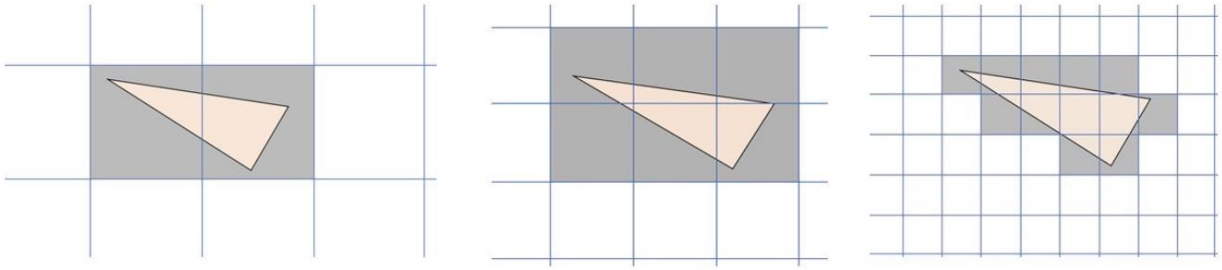
Géohachage – Geohash

https://www.ibm.com/docs/en/ias?topic=toolkit-geohashes

Geohash bit depth	North-South extent	East-West extent	Area
5	5,009,377.09	5,009,377.09	25,093,858,789,741.90
6	5,009,377.09	2,504,688.54	12,546,929,394,870.90
7	2,504,688.54	2,504,688.54	6,273,464,697,435.46
8	2,504,688.54	1,252,344.27	3,136,732,347,465.39
9	1,252,344.27	1,252,344.27	1,568,366,173,106.52
10	1,252,344.27	626,172.14	784,183,087,179.43
11	626,172.14	626,172.14	392,091,543,902.80
12	626,172.14	313,086.07	196,045,771,951.40
13	313,086.07	313,086.07	98,022,885,975.70
14	313,086.07	156,543.03	49,011,442,987.85
15	156,543.03	156,543.03	24,505,721,493.93
16	156,543.03	78,271.52	12,252,860,746.96
17	78,271.52	78,271.52	6,126,430,373.48
18	78,271.52	39,135.76	3,063,215,147.60
19	39,135.76	39,135.76	1,531,607,554.23
20	39,135.76	19,567.88	765,803,777.12
21	19,567.88	19,567.88	382,901,888.56
22	19,567.88	9,783.94	191,450,954.06
23	9,783.94	9,783.94	95,725,481.92
24	9,783.94	4,891.97	47,862,740.96
25	4,891.97	4,891.97	23,931,370.48
26	4,891.97	2,445.99	11,965,685.24
27	2,445.99	2,445.99	5,982,842.62
28	2,445.99	1,222.99	2,991,420.09
29	1,222.99	1,222.99	1,495,709.43
30	1,222.99	611.50	747,854.72
31	611.50	611.50	373,927.36
32	611.50	305.75	186,963.68
33	305.75	305.75	93,481.84
34	305.75	152.87	46,740.92
35	152.87	152.87	23,370.46
36	152.87	76.44	11,685.23
37	76.44	76.44	5,842.61
38	76.44	38.22	2,921.35
39	38.22	38.22	1,460.69
40	38.22	19.11	730.33
41	19.11	19.11	365.15
42	19.11	9.56	182.59
43	9.56	9.56	91.30
44	9.56	4.78	45.64
45	4.78	4.78	22.82

Couverture de géohachage

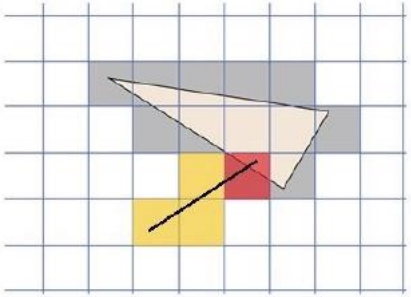
- Une **couverture de géohachage (geohash cover)** est un ensemble de cellules nécessaires pour couvrir une géométrie donnée



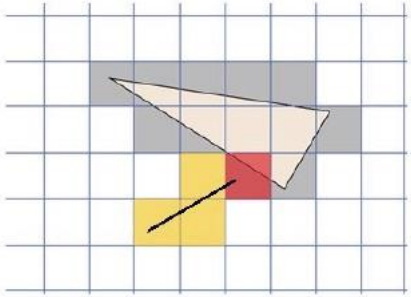
- Utilité : **filtrer** les géométries lors d'une requête de données géospatiales, afin **d'améliorer** les **performances**

Couverture de géohachage

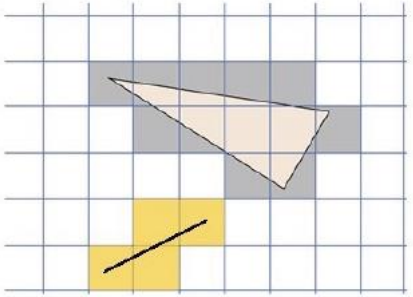
- Exemple : j'ai une table des routes et je souhaite connaître celles qui sont en intersection avec une parcelle de terrain donnée
  - Je commence par éliminer toutes les routes où les couvertures de géohachage ne partagent pas une cellule commune
  - Puis j'affine en appelant ST_INTERSECT pour les routes qui restent



La couverture de géohachage des 2 géométries partagent une cellule commune
→ je garde la route
→ puis ST_INTERSECT m'indique qu'il y a intersection



La couverture de géohachage des 2 géométries partagent une cellule commune
→ je garde la route
→ puis ST_INTERSECT m'indique qu'il n'y a pas d'intersection



La couverture de géohachage des 2 géométries ne partagent une cellule commune
→ je ne garde pas la route

Les fonctions de géohachage de DB2 for i

Db2 for i Enhancement	IBM i 7.5	IBM i 7.4	IBM i 7.3	IBM i 7.2
Geospatial Analytics	SF99950 Level 3	SF99704 Level 23	Not Supported	Not Supported

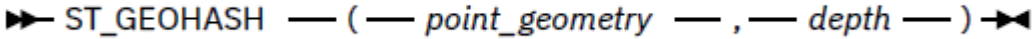
Fonctions de géohachage

- ST_FuzzyGeohashCover
- ST_FuzzyGeohashCoverExtend
- ST_Geohash
- ST_GeohashCover
- ST_GeohashCoverExtend

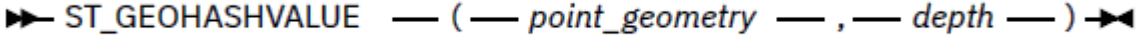
- ST_GeohashValue

Db2 for i Enhancement	IBM i 7.5	IBM i 7.4	IBM i 7.3	IBM i 7.2
Extend Watson Geospatial Analytics	SF99950 Level 4	SF99704 Level 25	Not Supported	Not Supported

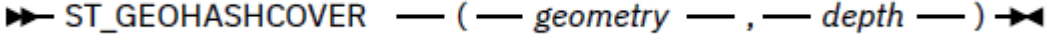
Les fonctions de géohachage de DB2 for i



- Un objet de type **ST_POINT** et une **profondeur** en paramètres
- Retourne une table avec une colonne (nommée GEOHASH, de type BIGINT), contenant la valeur du **geohash** du point – UDTF

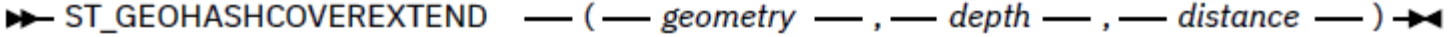


- Un objet de type **ST_POINT** et une **profondeur** en paramètres
- Retourne la valeur du **geohash** du point – UDF

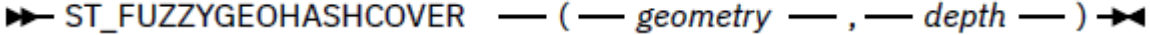


- Une **géométrie** et une **profondeur** en paramètres
- Retourne une table avec une colonne (nommée GEOHASH, de type BIGINT), contenant les **geohash** des cellules qui couvrent la géométrie – UDTF

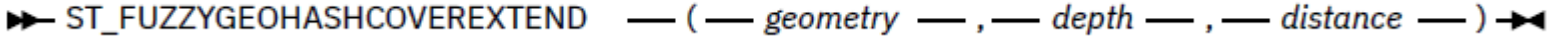
Les fonctions de géohachage de DB2 for i



- Une **géométrie**, une **profondeur** et une **distance** en paramètres
- Retourne une table avec une colonne (nommée GEOHASH, de type BIGINT), contenant les **geohash** des cellules qui couvrent la géométrie, "augmentées" par la distance (les cellules d'origine + une zone "tampon" autour de ces cellules) – UDTF



- Une **géométrie** et une **profondeur** en paramètres
- Retourne une table avec une colonne (nommée GEOHASH, de type BIGINT), contenant les **geohash** des cellules qui couvrent le MBR (*) de la géométrie – UDTF

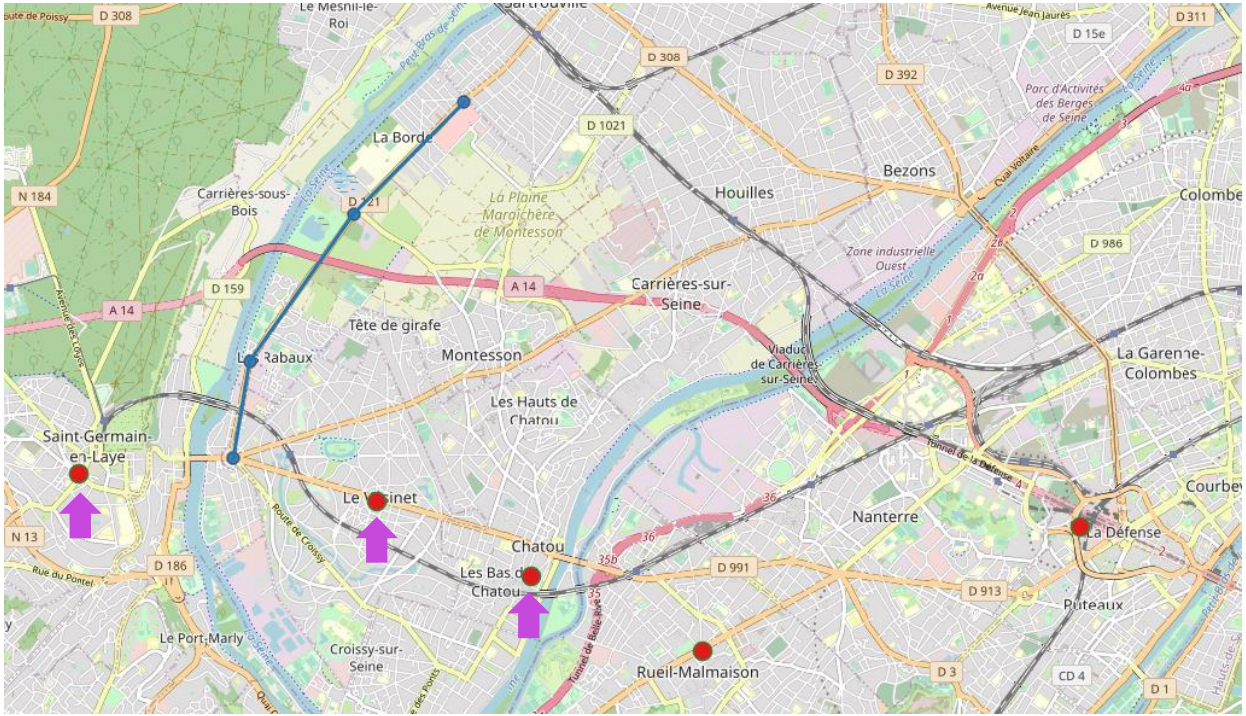


- Identique à ST_GEOHASHCOVEREXTEND mais avec le MBR (*) de la géométrie

(*) MBR : Minimum Bounding Rectangle : le rectangle minimal de délimitation d'une géométrie est le rectangle qui entoure et délimite la géométrie, et qui est formé par les coordonnées minimales et maximales (X,Y) de la géométrie

Géohachage – Exemple

- Quels sont les cinémas qui sont à moins de 5 km d'une route donnée ?



Géohachage – Exemple

- Quels sont les cinémas qui sont à moins de 5 km d'une route donnée ?

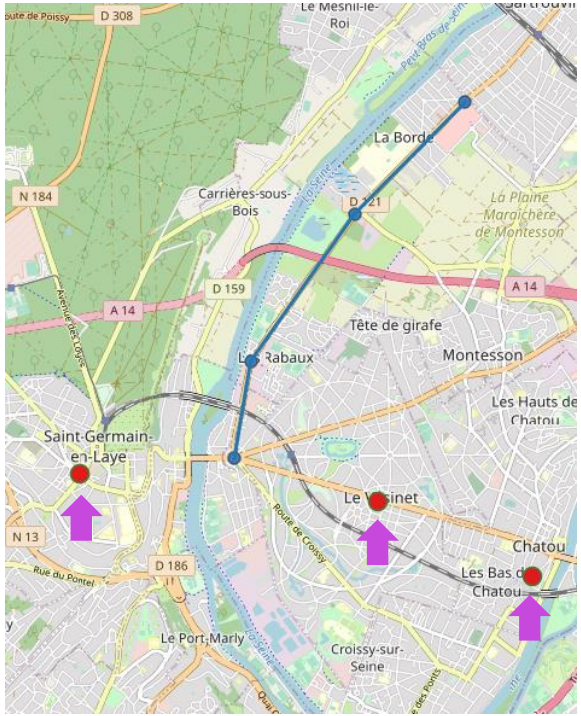
```
-- Alimentation d'une variable avec les coordonnées de la route

CREATE VARIABLE route_D121 QSYS2.ST_LINESTRING;

SET route_D121 = QSYS2.ST_LINESTRING('linestring(2.11373 48.89805,
                                                 2.11607 48.90717,
                                                 2.13097 48.92105,
                                                 2.14679 48.93153)');
```

Géohachage – Exemple

- Quels sont les cinémas qui sont à moins de 5 km d'une route donnée ?

```
386 -- Quels sont les cinémas qui sont à moins de 5 km d'une route donnée ?
387 -- 1. SANS GEOHACHAGE
388
389 SELECT nom, ville,
390        ROUND((QSYS2.ST_DISTANCE(route_D121, emplacement)/1000), 1)
391         AS "Distance en km"
392  FROM cinemas
393  WHERE QSYS2.ST_DISTANCE(route_D121, emplacement) <5000;
```

NOM	VILLE	Distance en km
Louis Jouvet	Chatou	3.4
Jean Marais	Le Vésinet	1.6
C2L	Saint-Germain en Laye	1.6



Géohachage – Exemple

- Quels sont les cinémas qui sont à moins de 5 km d'une route donnée ?

```
ALTER TABLE cinemas ADD COLUMN geohash_emplacement FOR COLUMN geoh_emplt BIGINT;
```

Nom de colonne	Nom de système	Type de données
IDCINE	IDCINE	INTEGER
NOM	NOM	VARCHAR
ADRESSE	ADRESSE	VARCHAR
CODE_POSTAL	CODPOS	VARCHAR
VILLE	VILLE	VARCHAR
EMPLACEMENT	EMPLACMT	QSYS2.ST_POINT
GEOHASH_EMPLACEMENT	GEOH_EMPLT	BIGINT

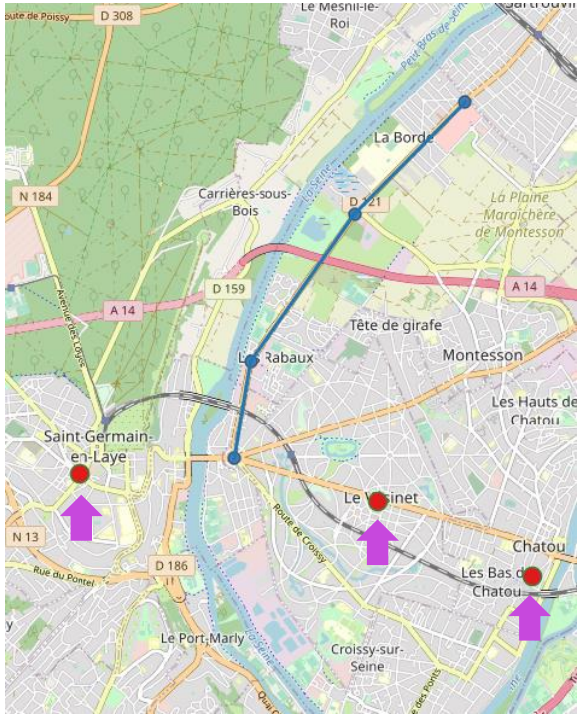
```
UPDATE cinemas SET geohash_emplacement =
                   QSYS2.ST_GEOHASHVALUE(emplacement, 25)
 WHERE geohash_emplacement IS NULL;
```

Géohachage – Exemple

- Quels sont les cinémas qui sont à moins de 5 km d'une route donnée ?

```
396  -- Quels sont les cinémas qui sont à moins de 5 km d'une route donnée ?
397  -- 2. AVEC GEOHACHAGE
398
399  WITH routeD121_geohash AS (SELECT geohash FROM
400        TABLE(QSYS2.ST_FUZZYGEOHASHCOVEREXTEND(route_D121, 25, 5000)))
401  SELECT nom, ville,
402         ROUND((QSYS2.ST_DISTANCE(route_D121, emplacement)/1000), 1)
403          AS "Distance en km"
404  FROM cinemas c JOIN routeD121_geohash r ON c.geohash_emplacement = r.geohash
405  WHERE QSYS2.ST_DISTANCE(route_D121, emplacement) <5000;
```

NOM	VILLE	Distance en km
Louis Jouvet	Chatou	3.4
Jean Marais	Le Vésinet	1.6
C2L	Saint-Germain en Laye	1.6



Autres possibilités – Utilisation directe des coordonnées

- Table des clients

Nom de colonne	Nom de système	Type de données	Longueur
IDCLIENT	IDCLIENT	INTEGER	
SOCIETE	SOCIETE	VARCHAR	50
ADRESSE	ADRESSE	VARCHAR	100
CODE_POSTAL	CODPOS	VARCHAR	5
VILLE	VILLE	VARCHAR	50
LONGITUDE	LONGITUDE	VARCHAR	20
LATITUDE	LATITUDE	VARCHAR	20
EMPLACEMENT	EMPLACMT	QSYS2.ST_POINT	

```
154 SELECT societe, adresse, code_postal, ville, longitude, latitude, QSYS2.ST_ASTEXT(emplacement) AS coordonnees FROM clients;
155
```

SOCIETE	ADRESSE	CODE_POSTAL	VILLE	LONGITUDE	LATITUDE	COORDONNEES
Le Cap	7 rue du Bas de la Plaine	78500	Sartrouville	2.16447	48.92703	POINT (2.16447 48.927029999999995)
Accro Camp	Côte de la Jonchère	92500	Rueil Malmaison	2.15511	48.86802	POINT (2.15511 48.868019999999994)
Golf de l'Ile Fleurie	Ile des Impressionnistes	78420	Carrières sur Seine	2.17350	48.90330	POINT (2.1734999999999998 48.903299999999994)
Stade Olympique Yves du Manoir	12 rue François Faber	92700	Colombes	2.24797	48.93187	POINT (2.24797 48.931869999999996)
Paintball 75	Ile de la Loge	78380	Bougival	2.11416	48.88278	POINT (2.11416 48.88278)
Entente Sportive Versaillaise	3 bis rue Champ Lagarde	78000	Versailles	2.14237	48.80168	POINT (2.14237 48.80168)
Piscine du Jardin Parisien de Clamart	37 rue du Dr Roux	92140	Clamart	2.26021	48.79656	POINT (2.26021 48.79656)

Autres possibilités – Utilisation directe des coordonnées

- Quels sont mes **clients** qui sont en **zone inondable** ? (d'après l'AZI : Atlas des Zones Inondables) → API Géorisques

https://www.georisques.gouv.fr/api/v1/gaspar/azi?latlon=2.17350%2C48.90330&rayon=1

JSON Données brutes En-têtes

Enregistrer Copier Tout réduire Tout développer Filtrer le JSON

```
results:                1
page:                   1
total_pages:            1
data:
  0:
    code_national_azi:  "93DDT19990004"
    libelle_azi:        "La Seine"
    liste_libelle_risque:
      0:
        num_risque:     "140"
        libelle_risque_long: "Inondation"
```

Est en zone inondable

https://www.georisques.gouv.fr/api/v1/gaspar/azi?latlon=2.26021%2C48.79656&rayon=1

JSON Données brutes En-têtes

Enregistrer Copier Tout réduire Tout développer Filtrer le JSON

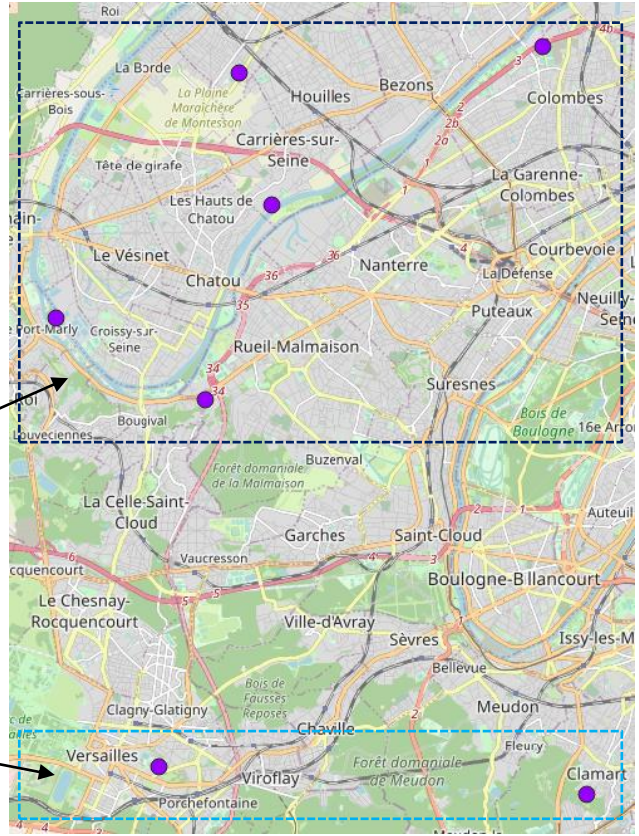
```
results:        0
page:           0
total_pages:    0
data:           []
response_code:  200
message:        null
next:           null
previous:       null
```

N'est pas en zone inondable

Autres possibilités – Utilisation directe des coordonnées

```
182 SELECT societe, ville,
183 (SELECT libelle_inondation FROM
184  JSON_TABLE(QSYS2.HTTP_GET('https://www.georisques.gouv.fr/api/v1/gaspar/azi?latlon='
185                            CONCAT(longitude)
186                            CONCAT '%2C'
187                            CONCAT(latitude)
188                            CONCAT '&rayon=1', options_WS),
189              '$' COLUMNS(libelle_inondation VARCHAR(50)  PATH '$.data[0].libelle_azi'  )))
190 FROM clients;
```

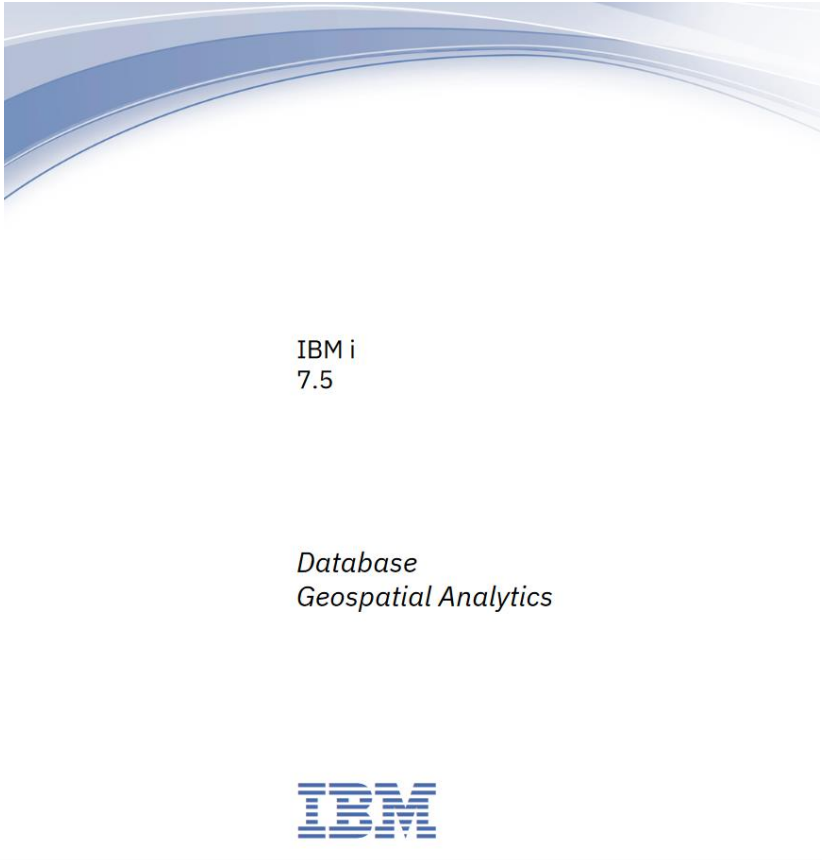
SOCIETE	VILLE	LIBELLE_INONDATION
Le Cap	Sartrouville	La Seine
Accro Camp	Rueil Malmaison	La Seine
Golf de l'Ile Fleurie	Carrières sur Seine	La Seine
Stade Olympique Yves du Manoir	Colombes	La Seine
Paintball 75	Bougival	La Seine
Entente Sportive Versaillaise	Versailles	-
Piscine du Jardin Parisien de Clamart	Clamart	-



4. Pour aller plus loin

Analytique géospatiale et DB2 for i

- Documentation
  - https://www.ibm.com/docs/en/i/7.5?topic=database-geospatial-analytics



90