Corrigé du TP2

Application de base de données spatiale

K. Zeitouni

## I) Création et requêtes d’une table jouet de villes

-- Creation table

CREATE TABLE city (

city\_id NUMBER PRIMARY KEY,

city\_name VARCHAR2(25),

latitude NUMBER,

longitude NUMBER);

-- Insertion "Munich"

insert into city values (1, 'Munich', 11.5174, 48.17571);

-- Insertion "Vienne"

insert into city values (2, 'Vienne', 16.3688, 48.20254);

-- Insertion "San Francisco"

insert into city values (3, 'San Francisco', -122.4194200, 37.7749300 );

-- Insertion "London"

insert into city values (4, 'Londres', -0.15307, 51.54931);

-- ajout attribut spatial

ALTER TABLE city ADD (loc SDO\_GEOMETRY);

-- mise à jour des données spatiales en utilisant les valeurs latitude et longitude.

UPDATE city SET loc = SDO\_GEOMETRY(2001, 8307, SDO\_POINT\_TYPE (LONGITUDE, LATITUDE, NULL), NULL, NULL);

-- mise à jour des metadata.

INSERT INTO user\_sdo\_geom\_metadata VALUES (

'city', 'LOC', SDO\_DIM\_ARRAY(

SDO\_DIM\_ELEMENT('Longitude',-180,180,0.5),

SDO\_DIM\_ELEMENT('Latitude',-90,90,0.5)

), 8307 );

-- Creation de l'index spatial.

CREATE INDEX city\_points\_spatial\_idx on city(LOC)

INDEXTYPE IS MDSYS.SPATIAL\_INDEX;

-- distance de "Munich" aux autres villes avec SDO\_GEOM.SDO\_DISTANCE

select b.city\_name, sdo\_geom.sdo\_distance( a.loc, b.loc, 1, 'unit=km' ) -- 1 de tolérance

from city a, city b

where a.city\_name='Munich' ;

-- calcul de distances entre les villes deux à deux

select a.city\_name as from\_city, b.city\_name as to\_city, sdo\_geom.sdo\_distance(

a.loc, b.loc, 1, 'unit=km' ) dist

from city a, city b

order by 1, 2; -- pour suivre l’ordre alphabétique de from\_city, to\_city

-- Version sans doubles (truc de < pour ne garder qu’une paire si symétrie)

select a.city\_name as from\_city, b.city\_name as to\_city, sdo\_geom.sdo\_distance(

a.loc, b.loc, 1, 'unit=km' ) dist

from city a, city b

where a.city\_name < b.city\_name

order by 1, 2;

-- version matrice de distances avec la clause SQL PIVOT

select from\_city, "Londres", "Munich", "San Francisco", "Vienne" from

(

select a.city\_name as from\_city, b.city\_name as to\_city, sdo\_geom.sdo\_distance(

a.loc, b.loc, 1, 'unit=km' ) dist

from city a, city b

)

pivot (sum(dist) for to\_city in (

'Munich' as "Munich",

'San Francisco' as "San Francisco",

'Vienne' as "Vienne",

'Londres' as "Londres"

) )

order by 1;

-- ou encore : (le select imbriqué peut être remplacé par une vue créée au préalable

select \* from (

\_ select a.city\_name as from\_city, b.city\_name as to\_city,

sdo\_geom.sdo\_distance( a.loc, b.loc, 1, 'unit=km' ) dist

from city a, city b where a.city\_name < b.city name )

pivot (sum(dist) for to\_city in (

'Munich' as "Munich",

'San Francisco' as "San Francisco",

'Vienne' as "Vienne",

'Londres' as "Londres" ) )

order by 1;

## II) Cas d’étude

Se référer à l’espace du cours pour télécharger les données sources : (1) dossier pour arrondissements qui contient plusieurs fichiers complémentaires du format shapefile ; fichiers texte simples pour les données « colleges » et « velib ».

Rem : Vous pouvez sauter la 1ère question. **- Les tables sont disponibles dans le schéma zeitouni.arrondissement, zeitouni.college, zeitouni.velib.**

1. a. Chargement des données fournies au format shapefile :

**A l'IUT :**

java -classpath C:\sqldeveloper\jdbc\lib\ojdbc8.jar;C:\sqldeveloper\sqldeveloper\lib\sdoutl.jar;C:\sqldeveloper\sqldeveloper\lib\sdoapi.jar oracle.spatial.util.SampleShapefileToJGeomFeature -h santorin -p 1521 -s info -u **votre\_compte** -d **votre\_MDP** -t arrondissement -f C:\**le-bon-chemin**\arrondissements\arrondissements -r 8307 -g loc

-- la commande ci-dessus met à jour les métadonnées. N’oubliez pas de regénérer l’affichage des tables (clic droit sur tables » et vérifier les données. On peut aussi vérifier sir les métadonnées et l’index ont été créés en cliquant sur spatial.

1. Chargement des données fournies au format text - suivre les étapes comme suit :

Importer Colleges via sqldeveloper.

D'abord changer le caractère decimal dans la session:

Clic droit sur tables sous le disque de votre connexion ->importer -> choisir le fichier colleges.txt, changer le délimiteur en ; et l'encodage en UTF8 ->suivant -> garder "insérer" dans méthode d'import, donner le nom de la table "college" -> suivant -> désélectionner la 1ère colonne puis celles après code INSEE et avant lat (juste ce qui nous intéresse) -> suivant

les nom de colonne ne doivent pas comporter d'espaces - rebaptiser les colonnes "Libellé" "Code\_INSEE" - vérifier que lat et lon sont bien NUMBER (sinon refaire correctement la commande ALTER SESSION)

Le chargement arrive à la fin. Pour vérifier : clic droit sur tables et regénérer (ce qui affiche les nouvelles tables). clic chaque table et afficher les données.

Maintenant on crée un sdo\_geometry pour college et velib - rajouter colonne de type SDO\_geometry puis la mettre à jour avec lat et lon :

ALTER TABLE college ADD (loc SDO\_GEOMETRY);

update college set loc = sdo\_geometry (

2001, -- type point 2D

4326, -- SRID

sdo\_point\_type(lon, lat, null),null,null);

--insérer dans les metadonnées (si Oracle 12c mais pas dans version 21c)

INSERT INTO USER\_SDO\_GEOM\_METADATA

VALUES (

'college',

'loc',

MDSYS.SDO\_DIM\_ARRAY(

MDSYS.SDO\_DIM\_ELEMENT('X', -180,180, 0.005),

MDSYS.SDO\_DIM\_ELEMENT('Y', -90, 90, 0.005)

),

4326 -- SRID doit être identique à la création de la geom

);

Créer Index spatial :

CREATE INDEX college\_idx

ON college(loc)

INDEXTYPE IS MDSYS.SPATIAL\_INDEX;

Refaire ces même étapes (import, ajout attribut de type geometry et mise à jour de cet attribut) pour les données velib.

1. Donnez le nombre de stations Velib par arrondissement.

select a.c\_ar, count(\*) nb\_stations from arrondissement a, velib v

where sdo\_inside (v.loc, a.loc)='TRUE'

group by a.c\_ar

order by 2 desc;

1. Donnez pour chaque arrondissement le nombre de collèges ayant le même code (numéro) d’arrondissement (Indication : Utilisez LIKE)

select a.c\_ar, count(\*) nb\_colleges from arrondissement a, college c

where c.arrondissement like a.c\_ar ||'ème%'

group by a.c\_ar

order by 1;

Autre solution par requête spatiale (devinez laquelle est plus couteuse):

select a.c\_ar, count(\*) nb\_colleges from arrondissement a, college c

where sdo\_inside (c.loc, a.loc)='TRUE'

group by a.c\_ar

order by 2 desc;

1. Vérifiez que ce code postal associé à un collège dans la table est géométriquement correcte.

SELECT c.libellé, c.arrondissement, a.c\_ar, c.code\_insee, a.c\_arinsee

FROM arrondissement a, college c

WHERE SDO\_RELATE(A.loc, c.loc, 'mask=anyinteract') = 'TRUE';

-- autre solution plus rapide car opérateur optimisé (il existe d’autres solutions:

SELECT c.libellé, c.arrondissement, a.c\_ar, c.code\_insee, a.c\_arinsee

FROM arrondissement a, college c

where sdo\_inside (c.loc, a.loc)='TRUE';

1. Donnez pour chaque collège la station Velib la plus proche et affichez la distance qui les sépare (à vol d’oiseau)

-- le dernier paramètre de SDO\_NN va de pair avec le retour de distance.

select c.libellé, v.nom, SDO\_NN\_DISTANCE(1) dist

from college c, velib v

where SDO\_NN(c.loc, v.loc,'sdo\_num\_res=1', 1) = 'TRUE' ORDER BY dist ;

1. Trouvez **le** collège le mieux desservi par des Velib (dont la distance moyenne à ses 3 plus proches stations Velib est la plus courte).

-- se traduit par une requête top k avec k=1.

-- On commence par la distance à la plus proche station :

select \*

from (select c.libellé, v.nom, SDO\_NN\_DISTANCE(1) dist

from college c, velib v

where SDO\_NN(c.loc, v.loc,'sdo\_num\_res=1', 1) = 'TRUE' ORDER BY dist)

where rownum = 1;

-- On affine avec la moyenne de distance aux 3 plus proche stations :

select \*

from ( **select libellé, AVG(dist) moy\_aux\_3PPV**

from (select c.libellé, SDO\_NN\_DISTANCE(1) dist

from college c, velib v

where SDO\_NN(c.loc, v.loc,'sdo\_num\_res=**3**', 1) = 'TRUE' )

group by c.libellé

ORDER BY **moy\_aux\_3PPV**)

where rownum = 1;

1. Fusionnez les arrondissements de Paris pour générer le contour de la ville

-- génère une seule ligne et une seule geométrie

SELECT SDO\_AGGR\_UNION(

SDOAGGRTYPE(a.loc, 0.005))

FROM arrondissement a;

1. Visualisez les arrondissements ayant le plus de stations Velib.

-- combinaison de prédicat topologique inside et requête top k.

select \*

from (select a.c\_ar, count(\*) nb\_stations from arrondissement a, velib v

where sdo\_inside (v.loc, a.loc)='TRUE'

group by a.c\_ar

order by 2 desc)

where rownum = 1;