Systèmes d'Informations Géographiques - SGBD et les extensions applicables à la gestion d'informations spatiales

Institut National des Sciences Appliquées - Rouen
Département Architecture des Systèmes d'Information
michel.mainguenaud@insa-rouen.fr

Principes

- Opérateur vs prédicat
  - Opérateur : donne le résultat en fonction de la sémantique (intersection, adjacence, …)
  - Prédicat : renvoie un résultat de type Vrai/Faux
- Extension du langage
  - Opérateur : clause Select
  - Prédicat : clause Where
- Problématique principale : Composition

Rappels SQL

- Base
  - Logique du 1er ordre sans fonction
  - Algèbre relationnel (union, minus, intersect)
  - Calcul relationnel de tuples (join, selection, ...)
- Principe de fermeture des opérateurs
  - Domaine de base : Relation
  - Opérateurs : Relation [x Relation] → Relation
- Extension avec des fonctions
  - Min, Max, Count, Avg, Sum
  - Ne renvoie qu'un seul résultat (1 tuple, 1 attribut)

Problème des fonctions

- Cohérence globale
  - Select From Where → relation
- Clause Select
  - Select min (degre) from Vins;
- Clause Where
  - Select *

    from Vins

    where deg > Select avg (degre) from Vins
- Comparaison attribut / relation

Problème des Dépendances Fonctionnelles

- DF :
  - $A \rightarrow B$: à une valeur de A ne correspond qu'une seule valeur de B
  - 3FN : pas de DF entre attributs non clé, pas de DF entre sous-ensemble de la clé et attribut non clé
- Opération de Groupement
  - En cas de jointure la clé d'une des relations n'est plus la clé de la relation obtenue après jointure (en général)
  - Ne peut apparaître dans la clause Select que des attributs qui apparaissent dans la clause Group By si cette dernière est présente

Exemple

- R1 : Buveurs (NB, nom, ville) => NB → Ville
- R2 : Recoltes (NB,NV)
- R3 : Vins (NV, degre)
- Requête
  - Select B.NB, B.ville

    From Buveurs B, Recoltes R, Vins V

    Where V.degre > 12 and B.NB = R.NB and R.NV = V.NV

    Group by B.NB, B.ville

    Having count(*) > 5

Raisons

- On veut garantir qu'une seule ville pour un buveur pour former un tuple à partir du partitionnement horizontal
  - Attribut de la relation : DF mais non gérée par le SGBD
- Quid si l'attribut ne fait pas partie de la relation de l'attribut faisant le partitionnement ?
  - Attribut hors relation : aucune garantie => apparaît dans le Group By

Extensions spatiales

- Lien sémantique : alphanumérique / spatial
- Nouveaux domaines → nouveaux opérateurs
- Quel formalisme pour le langage ?
  - Approche de type algébrique
    - Relation [x Relation] → Relation
  - Approche de type calcul (retenue dans la norme)
    - Attribut [x Attribut] → Attribut

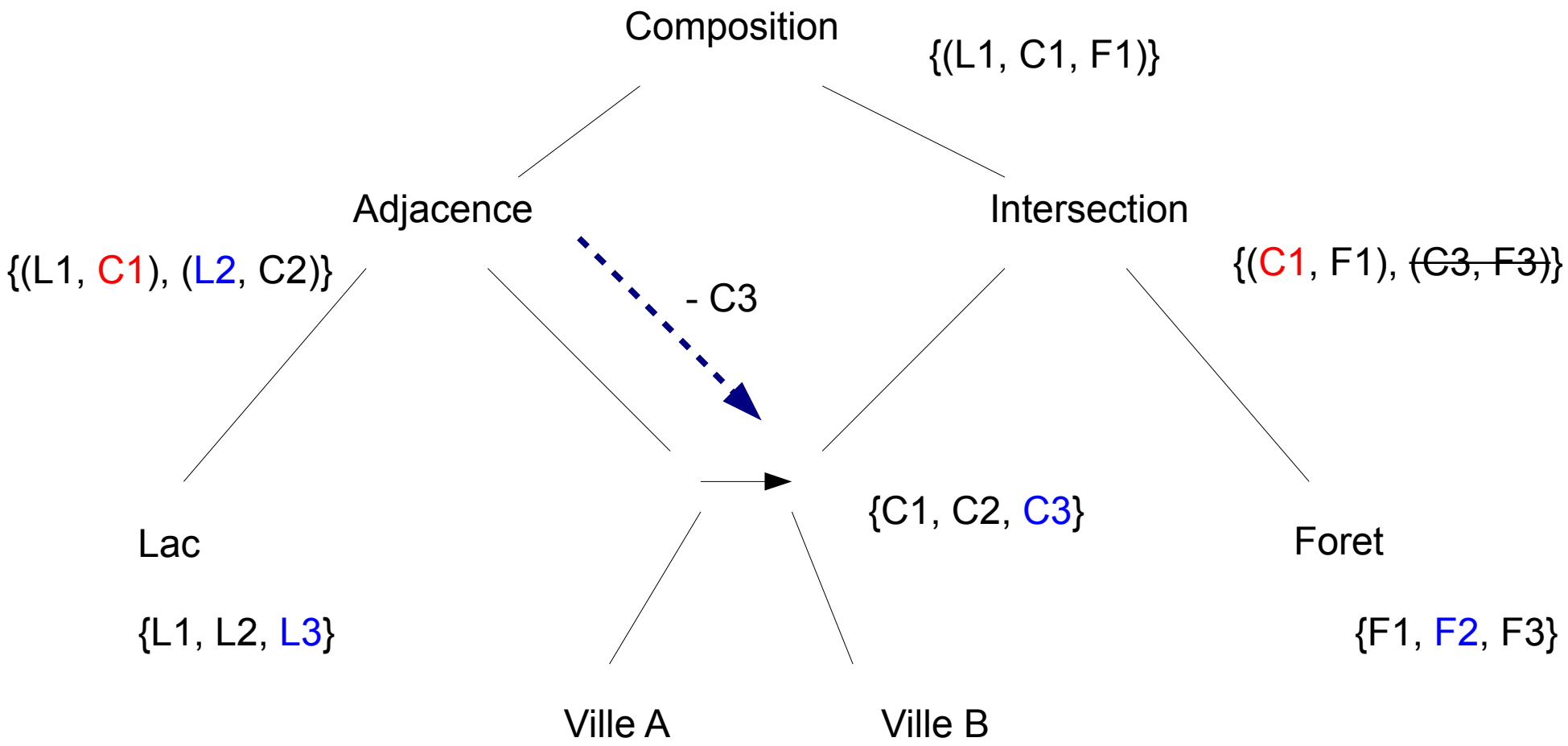
Objectifs

- Garantir la cohérence
  - Des informations spatiales
    - Opérateurs travaillant sur des représentations symboliques, sur des représentations réelles
  - Des information alphanumériques
    - Opérations d'agrégation / désagrégation
    - Sémantique spatiale associée au résultat de l'opérateur
  - Des liens entre les informations spatiales / alphanumériques après applications des opérateurs sur le résultat

Objectifs (2)

- Permettre la composition d'opérations
  - Soit dans la clause From : (Haas, Cody) → tuples
  - Soit dans la clause Select → tuples
- Conséquences
  - Rupture du premier ordre
  - Identification dans un langage fonctionnel algébrique des éléments identiques
  - Arbre algébrique vs. Graphe algébrique
    - Classique : pas de remise en cause des résultats intermédiaires (car propagation des attributs pour la projection)
    - Composition : remise en cause des résultats partiels

Exemple


Composition
{(L1, C1, F1)}
Adjacence
Intersection
{(L1, C1), (L2, C2)}
- C3
{(C1, F1), (C3, F3)}
{C1, C2, C3}
Lac
{L1, L2, L3}
Foret
{F1, F2, F3}
Ville A
Ville B

GML

- Dérivé de XML, proposé par Open Geospatial Consortium
- Dédié aux informations géographiques
  - Point (Point), ligne (LineString), ligne fermée (LinearRing), polygone (Polygon, Surface), courbe (Curve), fond de plan (Coverages), avec des trous (exterior/interior)...
- S'occupe du fond contrairement à KML (aussi proposé à l'OGC) qui fait forme et fond → engagement de cohérence à terme avec Google

Principes

- Objet = {Entités}
- Entité = {Propriétés}
- Collection d'entités = ensemble d'entités vues comme une seule entité + propriétés spécifiques
- Ensemble de schémas disponibles selon le domaine d'application (exemple : CityGML)
- Une même entité peut avoir des représentations différentes
- Transformation par XSLT

Organisation

- Sous typage de :
  - Entités
    - AbstractFeatureType
    - AbstractFeatureCollectionType
  - Géométrie
    - AbstractGeometryType
    - GeometryCollectionType
  - Propriétés
    - GeometryPropertyType
- Liens
  - features.xsd, geometry.xsd, xlink.xsd → schémas d'applications

Exemple

```
<Bridge>
   <span>100</span>
   <height>200</height>
   <gml:centerLineOf>                    ← ligne centrale du pont
         <gml:LineString>
            <gml:Point>
                <gml:pos>100 200</gml:pos>
            </gml:Point>
            <gml:Point>
                <gml:pos>200 200</gml:pos>
            </gml:Point>
         </gml:LineString>
    </gml:centerLineOf>
</Bridge>
```

<XXX xsi:schemaLocation="http://schemas.opengis.net/ XXX.xsd">

Principes des extensions (SQL2 / SQL/MM)

- Héritage
  - Approche classique
    - Vues : stockage des tuples au niveau des feuilles
    - Union / projection : stockage des tuples à tous les niveaux de la hiérarchie.
  - Approche par extension
    - Définition de deux relations dont une 'INHERITS' de l'autre
      - CREATE TABLE R1 (attributCommun domaine);
      - CREATE TABLE R2 (attributSupplémentaire domaine) INHERITS (R1);

Principes des extensions (3)

- Fonctions (select / where)
  - create function nom (paramètre(s)) returns domaine as

    'Requête SQL'

    language 'sql';
- Nouvelles manipulations (ST_XX pour le spatial)
  - Prédicats : ST_Intersects, …
  - Opérateurs : ST_Intersection, ...

Principes des extensions (2)

- Tuples (Select) :
  - A un niveau de la hiérarchie : from ONLY
  - Avec application récursive de la hiérarchie (défaut)
- Attribut
  - Par les fonctions spécifiques
  - Notation fonctionnelle : nomAttribut(<tuple>)

Un exemple : PostGIS

- Extension spatiale au dessus de PostgreSQL
  - Domaines
    - Open GIS : coordonnées planaires
      - GEOMETRY
    - Données géographiques : WGS 84 long/lat SRID 4326
      - GEOGRAPHY
- Open-Source : intégration de projets existants grâce à l'extensibilité de PostgreSQL
  - Proj4 : référentiels
  - Geos : opérateurs

Principes (2)

- psql -d <base> -f .../postgis.sql
  - Créer les fonctions, les opérateurs pour les manipulations spatiales, langages : SQL, C, …
  - Créer les relations, méta-données, pour la géométrie (geometry_columns) et les référentiels (spatial_ref_sys).
- psql -d <base> -f .../spatial_ref_sys.sql
  - Insérer les tuples dans la relation spatial_ref_sys pour gérer les référentiels disponibles (associés au SRID)

Principes (3)

- Norme : Compatible OGC « Simple features for SQL »
- Hors norme : une partie du 3D et du 4D
- SRID : Spatial Referencing IDentifier définit le système de coordonnées : WGS84, UTM, ...
- Formalisme
  - WKB : Well Known Binary
  - WKT : Well Known Text
  - GML : ST_asGML(the_geom)

WKT

- Exemples:
  - POINT(3.75 3.75) ou MULTIPOINT(0 0,1 2)
  - LINESTRING(1 2,12 8,12 5) ou MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))
  - POLYGON((0 0,0 4,4 4,4 0,0 0))
  - POLYGON((0 10,0 12,2 12,2 10,0 10),(0.5 10.5,0.5 11.5,1.5 11.5,1.5 10.5,0.5 10.5)) → avec trou
  - MULTIPOLYGON(

    ((6 4,6 10,8 10,8 4,6 4)),

    ((9 9,9 10,10 10,10 9,9 9)))
  - GEOMETRYCOLLECTION(

    POINT(2 3),LINESTRING(2 3,3 4))
  - ...

Déclaration

- Relation conventionnelle + attribut particulier
  - Nouveau type : geometry ou geography
    - Soit directement
      - create table villes (id int4, nom varchar(20), the_geom GEOMETRY);
    - Soit indirectement
      - AddGeometryColumn ([<schéma>], <relation>, <attribut>, <srid>, <type>, <dimension>)
      - Select AddGeometryColumn
        ('','ville','the_geom','-1','POLYGON',2);
    - Soit automatiquement
      - Shp2pgsql : génère un fichier xx.sql
        - Création d'une relation
        - Nouveau type indirect
        - Génération de « insert »

Insertion

- Par des « insert »
  - Avec géometrie codée
    - Shp2pgsql
      - Insert into villes (id, nom,the_geom) values (1, 'V1','010300000.....');
  - Avec transcodage
    - ST_GeometryFromText
      - insert into villes (id, nom, the_geom) values (1, 'V1', ST_GeometryFromText('POLYGON ((0 0, 0 4, 4 4, 4 0, 0 0))', -1));
- Par des « Copy from »

Insertion (2)

- Gestion de l'intégrité (absente par défaut)
  - Permanente

    Alter table <nomRelation>

    Add constraint <nomContrainte>

    Check (ST_IsValid(<nomAttributSpatial>));
  - Ponctuelle

    Select distinct ST_IsValid (the_geom) from <relation>;

Visualisation

- Illisible

  Select * from villes;

  id | nom | the_geom

  1 | V1 | 010300000 .....

- Par transcodage

  Select id, nom, ST_AsText(the_geom) from villes;

  id | nom | the_geom

  1 | V1 | POLYGON((0 0,0 4,4 4,4 0,0 0))

  → attention aux troncatures

- Par outils externes :
  - Viewer : pgsql2shp + Qgis, ST_AsGML, ...

Manipulations

- Méta - « Méta-like »
  - Environnement : PostGIS_<XX> ()

    Select PostGIS_Full_Version();
  - Structure spatiale

    Select <Clé>, ST_GeometryType(<attributSpatial>)

    From <relation>;

Manipulations (2)

- Base
  - Opérateurs : ST_<XX>
    - Sur la géométrie : (exemple : ST_Intersection)
    - Sur le rectangle englobant : (exemple &&)
  - Prédicats

    ST_Intersects (r.the_geom, v.the_geom)
  - Constructeurs

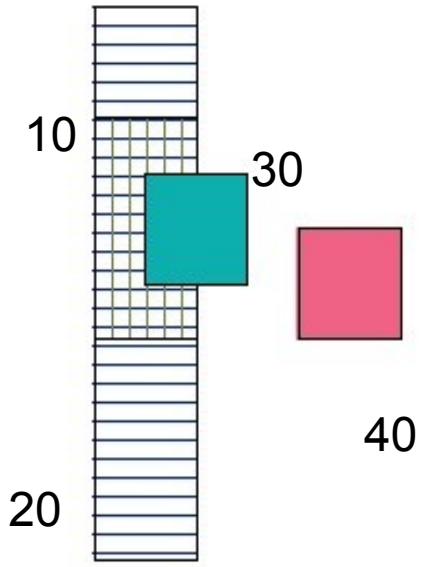
    ST_GeometryFromText(text WKT [, integer srid])
  - Autorise la composition d'opérateurs

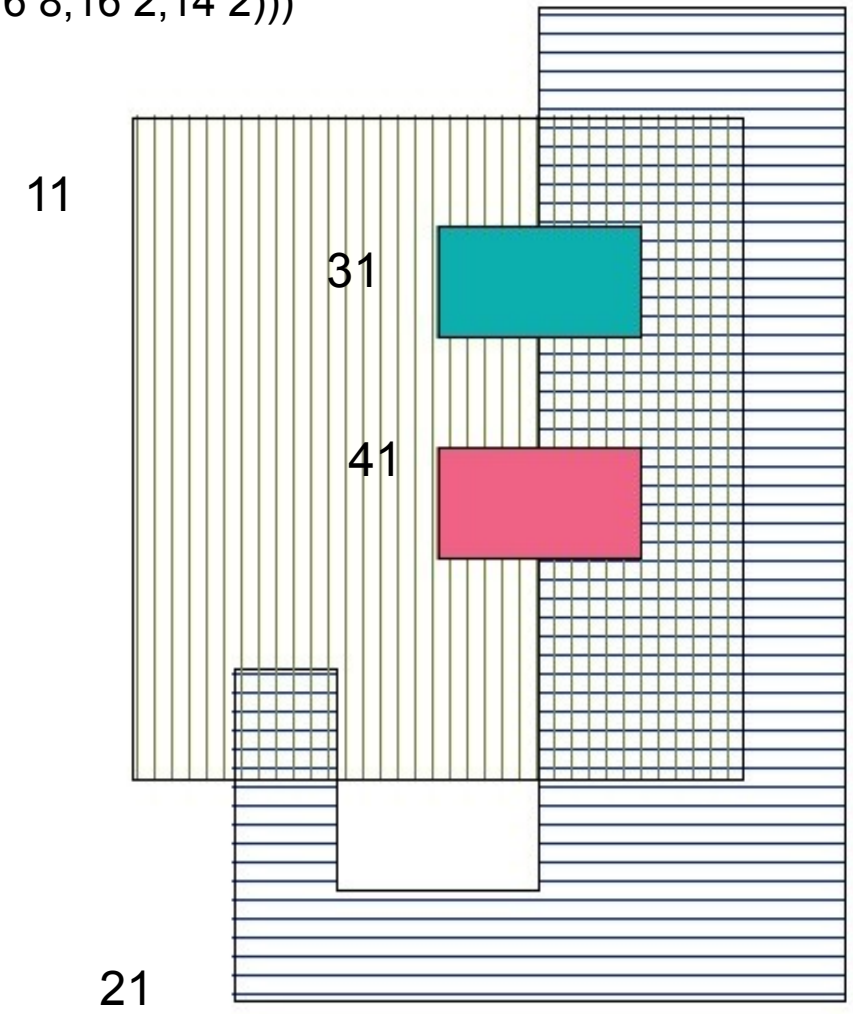
Exemple

```
Select A.id, B.id, C.id, D.id
        ST_Intersection (C.geom,
                         ST_Intersection (A.geom, B.geom))
        ST_Intersection (D.geom,
                         ST_Intersection (A.geom, B.geom)),
         ST_Intersection (A.geom, B.geom)
From   A, B, C, D
Where  ST_Intersects (A.geom, B.geom) and
         ST_Intersects (D.geom, ST_Intersection (A.geom,
                                                 B.geom)) and
         ST_Intersects (C.geom, ST_Intersection (A.geom,
                                                 B.geom));
```

id	id	id	id	interdinterab	intercinterab
11	21	31	41	POLYGON((14 5,15 5,15 4,14 4,14 5))	POLYGON((14 7,15 7,15 6,14 6,14 7))

interab
MULTIPOLYGON(((11 2,11 3,12 3,12 2,11 2)),((14 2,14 8,16 8,16 2,14 2)))





Positionnement du problème

- Approche algébrique
  - Opérateurs définis sur les relations
    - Alphanumérique
    - Spatial
  - Cohérence sémantique par définition de règles
  - Composition « difficile »
- Approche Calcul relationnel de tuple
  - Opérateurs au niveau de l'attribut « spatial »
  - Approche prise par Postgis
  - Risque d'incohérence sémantique car absence de règles sur les projections alphanumériques

Transactions (longues)

- Niveau SQL
  - Serializable transaction level (sécurité niveau maximal)

    Select EnableLongTransactions ();

    Select DisableLongTransactions ();
  - Vérification (trigger) :

    CheckAuth (<relation>, <nom de l'attribut clé>)
  - Demande d'autorisation pour cette transaction

    Select AddAuth (<user>);
  - Verrouillage

    Select LockRow (<relation>, <clé tuple>, <user>, [, <durée>]);
  - Déverrouillage :

    Select UnlockRows (<user>);

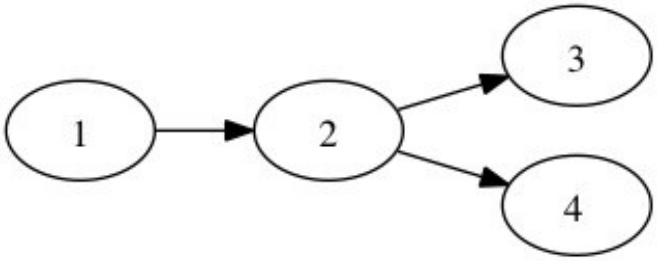
Utilisation de la récursivité

- Relation(s) modélisant des liens de type composant-composé, des réseaux de communication (orienté ou non orienté), ...
  - DAG (arbre, graphe sans cycle), graphe avec cycles
- Opérateurs d'accessibilité :
  - → (origine fixée, destination fixée) → PgRouting
  - → (origine fixée, destinations inconnues)
  - → (origines inconnues, destination fixée)
  - Avec ou sans contrainte agrégative

Jointure

- (Auto-)Jointure :
  - Rapprochement entre deux tables (resp. la même)
    - Reseau (id, origine, destination, cout)
  - Impose le niveau de profondeur
    - 1 seul niveau : (longueur = 2)

```
Select      distinct pere.origine, fils.destination
From        reseau pere, reseau fils
Where       pere.destination = fils.origine
```

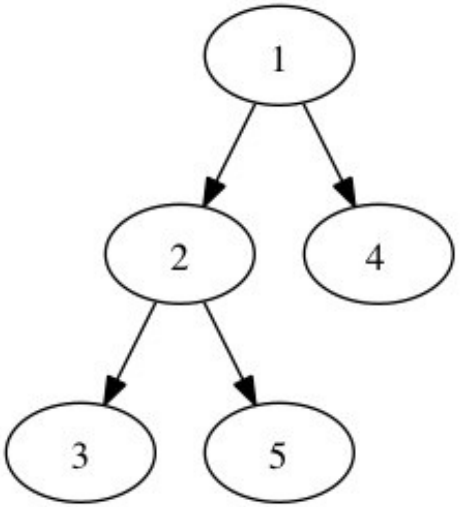


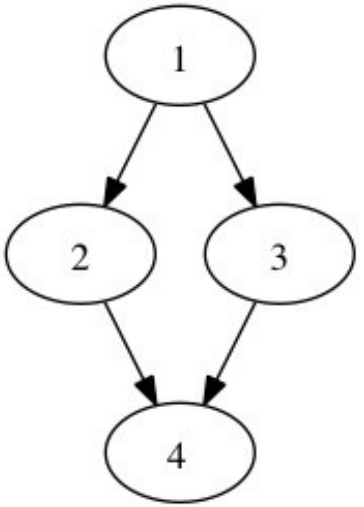
Reseau	id	origine	destination	cout
	#1	1	2	10
	#2	2	3	15

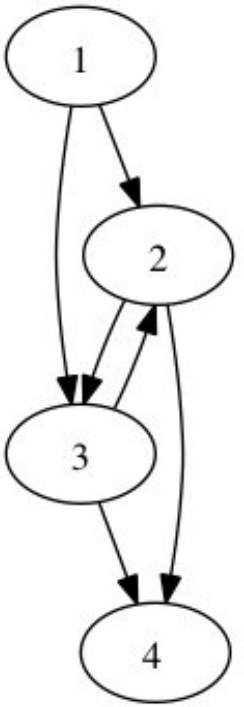
....

Récursion

- Récursion : a priori le niveau n'est pas connu
  - Sans cycle : arrêt assuré
    - Arbre : par définition il ne peut pas y avoir de double dans la solution
  - Avec cycle (en gérant l'historique) :
    - Manuellement (niveau de profondeur) : restrictif
    - SGBD







Principes

- Algorithmes : naïf, semi-naïf, magic-sets, …
- Langages : Prolog, Datalog, SQL
  - With recursive <RelationTemporaire> <schéma> as

    (

    Select <initialisation de la récursion>

    union

    Select <recursion sur la relation temporaire et la relation de base>

)

    Select <resultat final>

Exemple

- Reseau (gid integer, origine integer, destination integer)
- Tous les nœuds accessibles à partir de 1 :

```
with recursive deep (n) as (
        select      destination
        from        reseau
        where       origine = 1
    union
        select      destination
        from        reseau, deep
        where       reseau.origine = deep.n)
select * from deep;
```

PgRouting

- Calcul d'itinéraires à partir d'une origine et vers une destination
- Algorithmes
  - Dijkstra (sans heuristique) : shortest_path
  - A* (avec des heuristiques) : shortest_path_astar
  - Shooting star (restrictions sur les croisements) : shortest_path_shooting_star
  - Voyageur de commerce (TSP)
  - Isolignes sur les distances
- Double approche thématique / réseau

Création du réseau

- Directement par la structure :
  - Table routing (gid, nom, source, target, [cout, ...], [the_geom])
- Soit à partir d'une assignation
  - Schéma de relation :
    - Create table routing (gid integer, nom integer)
    - Select AddGeometryColumn('routing', 'the_geom', -1, 'GEOMETRY', 2);
    - Chargement de la géometrie
    - alter table routing add column source integer;
    - alter table routing add column target integer;
    - alter table routing add column cost float;
    - update routing set cost = ST_length(the_geom);
  - Génération d'un graphe non orienté basé sur la topologie (donc très (trop!) proche du physique)
    - SELECT assign_vertex_id
      ('<relation>', <tolérance>, '<attribut spatial>', '<attribut cle>');
    - SELECT assign_vertex_id('routing', 0.00001, 'the_geom', 'gid');

Spatialisation (latitude /Longitude)

- Absente : pas de A*
- Présente ou générée
  - ALTER TABLE routing ADD COLUMN x1 double precision;
  - ALTER TABLE routing ADD COLUMN y1 double precision;
  - ALTER TABLE routing ADD COLUMN x2 double precision;
  - ALTER TABLE routing ADD COLUMN y2 double precision;
  - UPDATE routing SET x1 = x(ST_startpoint(the_geom));
  - UPDATE routing SET y1 = y(ST_startpoint(the_geom));
  - UPDATE routing SET x2 = x(ST_endpoint(the_geom));
  - UPDATE routing SET y2 = y(ST_endpoint(the_geom));

Performances

- Création d'index sur l'origine et la destination
  - Create index <nom> on
    <relation> (<attribut origine/destination>) ;
  - Create index sourceIndex on routing(source) ;
  - Create index targetIndex on routing(target) ;
- Toujours se questionner sur l'utilité d'un index

Requête - chemin

- SELECT * FROM <nom algo>

  ('SELECT gid as id,

  source::integer,

  target::integer,

  cost::float

  [,x1, y1, x2, y2]

  FROM routing',

  <origine>, <destination>,

  <orienté ?>, <coût inverse ?>);

Isoline

- SELECT *

  FROM driving_distance('

  SELECT gid AS id,

  source::integer,

  target::integer,

  cost::float

  FROM routing',

  <depart>, <distance>,

  <orienté?>, <coût inverse ?>)

« TSP » … ?

- SELECT *

  FROM tsp('SELECT DISTINCT

  source as source_id,

  x1::double precision as x,

  y1::double precision as y

  FROM routing

  WHERE source IN

  (<noeuds visités>)',

  '<noeuds visités entre virgules>',

  <depart>);

Larges objets binaires

- SQL définit un BLOB (Binary Large Object)
  - PostgreSQL ne respecte pas la norme
  - Type de base : bytea
- Avantages
  - Homogénéisation du stockage (alphanumérique et raster-vidéo, …)
  - Disponibilité de l'environnement BD (sauvegarde, droits, …)
- Inconvénients
  - Pas d'opérations (programme externe ou procédure stockée)

Fonctionnement

- Stockage dans une relation système
  - pg_largeobject (loid oid, pageno integer, data bytea)
- Macro de (dé-)chargement
  - \lo_import ('<nom de fichier>')
  - \lo_export (<attribut oid>, '<nom fichier>')
  - Attention en SQL aux droits d'accès pour les fichiers : c'est le serveur qui travaille, pour le client accès via l'interface C par exemple.
- Effacement par \lo_unlink (<oid>)

Niveau client

- Création d'une relation avec :
  - Des attributs alphanumériques classiques
  - Un attribut de type « oid »
- Voir les larges objets existants : \lo_list
- Mettre en place des procédures de manipulation avec un langage interfacé avec la BD
  - C : #include "libpq-fe.h", #include "libpq/libpq-fs.h"
  - Manipulations « à la Unix » : lo_lseek, lo_read, ...

Rules (Règles)

- Pas spécifiques au spatial
- Principe effectuer des opérations en plus ou à la place d'un ordre SQL lors d'un insert, update, delete
- Create rule <name> as on <event> to <relation> do <also / instead> <ordre SQL>
- Exécution transparente à l'utilisateur
- Insert/update : utilisation du new.<attribut> (ou old)

Trigger / procédure stockée

- Pas spécifique au spatial
- Déclencher des ordres sur une opération sur une table (similaire aux règles)
- Create function <nom> () returns trigger as

  'texte' language '<langage>'
- Create trigger <nom> before insert on <table>

  for each row execute procedure <nom>() ;
- Exécution transparente à l'utilisateur
- Utilisation d'une structure d'échange ou du new/old (plpgsql)

Trigger / procédure stockée (2)

- Quand : Before, After + une clause when pour les conditions
- On <événement>
  - Insert, Update, Delete, Truncate : éventuellement une combinaison avec un 'or'
  - Update : peut être ciblé sur une ou plusieurs colonnes
  - Pas de select contrairement aux règles
- ATTENTION : For each
  - Row : un appel par tuple au trigger
  - Statement : un appel global

Langages - plpgsql

- Langage procédural classique
- Variables locales typées (section DECLARE)
- Ordres de traitement (section BEGIN /END)
  - Ordre SQL avec récupération de variables
  - Actions d'affectation, opérations, …
- Définition des NEW et OLD
- Retour de la fonction : TRIGGER

Langages (2) - C

- Utilisation de la SPI (Server Programming Interface)
- Structure de contexte (TriggerData)
  - Fonctions d'accès à l'environnement (mode before/after, ordre insert/update/delete, …)
  - Accès par la structure aux informations (nom de relation, position des attributs, valeur des attributs, …)
- Envoi d'ordres directement au SGBD