Introduction à PostGIS

Thomas Soubiran
Ceraps (UMR 8026 CNRS - Université de Lille)
https://numa.hypotheses.org

École d'été méthodes quantitatives en sciences sociales

Lille, 2 juillet 2021

Introduction

- présentation d'un troisième environnement pour travailler l'information géographique
- l'extension spatiale PostGIS
- qui ajoute le support d'objets géographiques dans le gestionnaire de base de données relationnelles PostgreSQL
- comme, p. ex., le packages `sf` dans `R`
- ou, auparavant, `sp(dep)`, `rgeos`,...

Introduction

- PostGIS propose une autre interface vers les mêmes librairies
- que celles utilisées par les autres environnement vus précédemment
  - `GDAL/OGR` (Geospatial Data Abstraction Library)

    lecture, écriture
  - PROJ

    projection cartographiques
  - dont le développement est coordonné et soutenu par l'Open Source Geospatial Foundation (OSGeo)
  - ainsi que `GEOS`

    géométrie : union, intersection,. . .
- mais cette fois pour le langage langage `SQL`
- qui est un langage spécialisé pour la manipulation de données

  comme `R` est un langage spécialisé pour l'analyse statistique

Note : les librairies sont souvent livrées avec des utilitaires en ligne de commande utile pour la préparation des données avant l'importation dans un environnement ou un autre

pour extraire des données ou changer la projection

Introduction

- comme les trois environnements utilisent les mêmes librairies
- ils proposent sensiblement les mêmes fonctionnalités
- mais de façon différentes
- QGIS a l'avantage de l'interactivité
- R permet d'utiliser p. ex. de nombreuses librairies statistiques

  comme Python ou d'autres environnements

PostgreSQL

- PostgreSQL permet le traitement de données volumineuses de façon efficace
- le traitement de l'information géographiques peut en effet nécessiter :
  - des opérations complexes

    comme l'intersection de géométries
  - sur des gros volumes de données
- sans que cela implique nécessairement de larges territoires
  - dépend aussi de la précision|résolution
  - et du nombre d'objets :

    Exemple : fond de carte des surfaces cultivées dans la Mel
    - 651 km^2
    - 13 015 polygones
    - 654 328 points
- c'est pourquoi les données IGNF sont souvent livrées au département

  plusieurs couches avec beaucoup de polygones et beaucoup de points

PostgreSQL

- PostgreSQL est un logiciel qui a une longue histoire
- car il s'inscrit dans une suite de projet différents
- Ingres —années 1970—
- Postgres —post-Ingres, années 1980—
- et enfin PostgreSQL —années 1990—
- notamment par Michael Stonebraker qui contribuer à développer le modèle des bases de données relationnelles orientées objet

Note : comme `R` mais différemment de `R`

- de nombreux autres RDBMS existent en dehors de PostgreSQL
- avec un support géographique

  Exemple : comme MySQL, SQLite,...

- et pas uniquement relationnelles

  NoSQL —Not Only SQL—

- qui peuvent eux aussi être utilisés à partir de `R`, `Python`,...
- **Note :** SQLite

  - SQLite est une alternative zéro–conf intéressante pour une utilisation locale
  - qui est utilisé p. ex. par QGIS ou `firefox`
  - et aussi à partir de `R`

PostgreSQL

- une des particularités de PostgreSQL est l'accent mis par ses développeurs sur l'extensibilité
- PostgreSQL propose une interface pour faciliter l'intégration d'extensions
- p. ex. pour étendre le type de données gérées
- comme PostGIS

  avec le type `geometry` pour les données géographique

- et des dizaines d'autres extensions
- mais pas de dépot officiel comme le `cran`

  mais voir notamment ce **site**

Indexation

- plus généralement, PostgreSQL fait en partie la même chose que, p. ex., R

  comme sélectionner des enregistrements répondant à différents critères ou faire de jointures entre des tables

- mais de façon plus efficace pour certaines d'entre elles
- la spécialisation du langage permet en effet de réaliser un ensemble d'optimisations
- notamment, par l'utilisation d'index de données dans l'exécution des requêtes

Indexation

- un index informatique est similaire à l'index d'un livre
- pour trouver un mot ou toute les occurrences d'un mot dans un ouvrage :
  - une sélection de mots sont triés alphabétiquement et cette liste renvoie aux pages du livres où apparaît le mot
  - au lieu d'une lecture séquentielle jusqu'à trouver l'occurrence recherchée ou jusqu'à la fin
- plusieurs indexes possibles pour un même livre :
  - *index nominum*
  - *index rerum*
  - *index verborum*

Indexation

- pour un fichier, c'est pareil
- pour chercher une valeur dans une colonne –ie : un vecteur
- on peut,
  - soit parcourir le vecteur jusqu'à trouver la valeur ou jusqu'au bout
  - ou générer un index
- qui va permettre d'accélérer la recherche
- comme les tables de hachage (condensat)

Tables de hachage

- les tables de hachage permettent de créer des indexes pour les structures de données de type table clef–valeur
- et on souhaite retrouver une valeur correspondant à une clef

Exemple : `COG`

clef	valeur
"01001"	"L'Abergement-Clémenciat"

on indexe le code commune pour retrouver le libellé de la commune

ou

clef	valeur
"L'Abergement-Clémenciat"	"01001"

on indexe le libellé de la commune pour retrouver le code commune

Fonction de hachage

- formellement, une fonction de hachage est
  - une fonction H sur une ensemble U
  - qui prend un élément de U pour lui associer un entier
  - dans un interval $[O, M)$ ou M est un entier

$$H : X \rightarrow [0, M)$$

- la fonction de hachage transforme donc la clef en nombre

quel que soit son type : entier, flottant, chaîne de caractère,. . .

- et ce nombre va servir d'index
- en donnant la position dans la table
- où la valeur va être stockée

Exemple

- Exemple de fonction de hachage avec R

```
h ← function(x, sz, init=5381, M=33){
  m ← 2^32
  h = init
  for( l in as.integer(charToRaw(x)) ){
    h ← ( (M * h) %% m + l ) %% m
  }
  as.integer(h %% sz)+1L
}
```

- avec sz le nombre d'éléments à insérer dans la table
- fonction seulement pour l'exemple, n'utilisez jamais quelque chose comme ça dans la vraie vie
  - à la fois cet algorithme
  - et R pour l'implémenter
  - d'autant plus qu'elle ne gère pas les collisions
- utilisez plutôt la fonction `match()`

```
match("01001" , commune2021$COM)
```

Exemple

- sz vaut 37742 pour le fichier commune2021 du `COG`
- on obtient alors
  - h("L'Abergement-Clémenciat", 37742) retourne 13938
  - h("L'Abergement-Clémenciat", 37742, 0, 31) retourne 36683
- on insére donc "01001" à la position 36683 —ou 13938—

index	valeur	clef
36682	"09167"	"Lézat-sur-Lèze"
36683	"01001"	"L'Abergement-Clémenciat"
36684	NA	NA
. . .	NA	NA
36686	"29240"	"Rosnoën"

- pour chercher le code commune de L'Abergement-Clémenciat
- il suffit de faire la même opération
- voir le fichier `cartogaphie/postgis/scripts.R` du **dépôt en ligne**

Hachage

- indexation et recherche sont donc symétriques
- au lieu de parcourir tout le fichier, il suffit d'appliquer la fonction aux clefs
- la fonction de hachage
  - permet de savoir si la clef est dans la table
  - ou de retrouver la valeur correspondant à la clef
  - comme pour les pages d'un livre

    pour l'index d'un livre, la fonction de hachage, c'est l'ordre alphabétique
- propriétés d'une fonction de hachage
  - uniformité du résultat
  - pour limiter les collisions qui ne manqueront pas d'arriver
  - principe des tiroirs (Dirichlet)

    Note : si n chaussettes occupent m tiroirs, et si $n > m$, alors au moins un tiroir doit contenir strictement plus d'une chaussette
  - différentes stratégies sont envisageables pour traiter le problème

Arborescences

- d'autres indexes utilisent une arborescence
- comme un arbre binaire
- exemple au tableau
- autres arborescences :
  - `arbre AVL (AVL tree)`
  - `arbre B (B-tree)`
  - `arbre bicolore (Red-black tree)`
  - `...`

Indexation spatiale

- les coordonnées géographiques sont représentées par deux variables numériques
  - longitude–latitude –coordonnées polaires—
  - ou x–y —coordonnées cartésiennes— pour les données projetées
- indexation multidimensionnelle
- pour laquelle il existe
  - des techniques générales
  - des techniques spécifiques pour des coordonnées géographiques
- PostGIS utilise l'indexation proposé par PostgreSQL

  GiST (Generalized Search Tree), généralisation de `B+ tree`

Modèles de données

- importance de comprendre le modèle de données utilisé par le logiciel que vous utilisez
  - c-à-d comment un logiciel organise les données
  - et quelles opérations sont permises par ce modèle
- y compris pour un tableur
- `R` repose notamment sur l'algèbre linéaire
- vecteur et opérations vectorisées :
  - **x**+**y** additionne chaque élément des vecteurs
  - **x**^2 élève chaque élément du vecteur à la puissance 2

    (**x**^2)==(**x***x) (ou plutôt, (**x**^2-**x***x)<2.220446e-16 car s'il s'agit de flottants)
  - `tolower`(m) change la casse de chaque chaîne de caractères du vecteur en minuscule

Le modèle relationnel

- PostgreSQL organise les données sous forme de « relations »

  c'est pourquoi PostgreSQL est un GBD dit relationnel

- relation :

  - ensemble —liste— de tuples —liste ordonnée—
  - $\simeq$ tables
  - $\simeq$ `data.frame`
  - mais orienté ligne dans PostgreSQL

    à la différence de `R` qui est orienté colonnes

- PostgreSQL repose donc sur le modèle relationnel
- qui est un modèle mathématique pour la gestion de base de données développé par Edgar F. Codd

  reposant lui-même sur la logique du premier ordre

- qui définit un ensemble d'opérations sur les relations
- qui, elles-mêmes, peuvent être appréhendées visuellement au moyen de la théorie des ensembles

  voir plus loin

Exemple

- opérations courantes
  - filtrer les données
  - fusionner les données
  - transformer les données
- **Exemple :** données électorales au bureau de vote
  - table à la commune : nombre d'inscrits, de votants, d'exprimés, de blancs et de nuls
  - table pour les listes ou candidats : suffrages
- pour calculer la proportions des votes parmi les votants ou les inscrits
- il faut fait correspondre le nombre d'inscrits aux différents suffrages
  - càd fusionner les deux tables
  - en utilisant plusieurs identifiants

    code commune, année, élection, tour

Exemple

- pour les résultats à Lille au 2e tour des régionales, il faut
  - filtrer sur l'élection, le tour, et la municipalité
  - fusionner les tables communes et suffrages
  - diviser les suffrages par le nombre d'inscrits
- table commune

COM	an	elec	tour	inscrits	votants	exprimés	votes blancs	votes nuls
59350	2021	'REG'	2	124 914	37 554	36 583	599	372

- table suffrage

COM	an	elec	tour	liste	suffrages
59350	2021	'REG'	2	Liste Karima DELLI	13 244
59350	2021	'REG'	2	Liste Xavier BERTRAND	19 172
59350	2021	'REG'	2	Liste Sébastien CHENU	4 167

- résulat attendu

COM	an	elec	tour	liste	suffrages	inscrits	exprimés	pinscr	pexp
59350	2021	'REG'	2	Liste Karima DELLI	13 244	124 914	36 583	.11	.36
59350	2021	'REG'	2	Liste Xavier BERTRAND	19 172	124 914	36 583	.15	.52
59350	2021	'REG'	2	Liste Sébastien CHENU	4 167	124 914	36 583	.03	.11

Représentation alternative

- dans ce cas, les données sont stockées dans des tables
- mais d'autres modèles de données sont envisageables
- comme une arborescence

  modèle mathématique : la théorie des graphes

- hiérarchie

  $\textit{élection} \rightarrow \textit{année} \rightarrow \textit{tour} \rightarrow \textit{commune}$

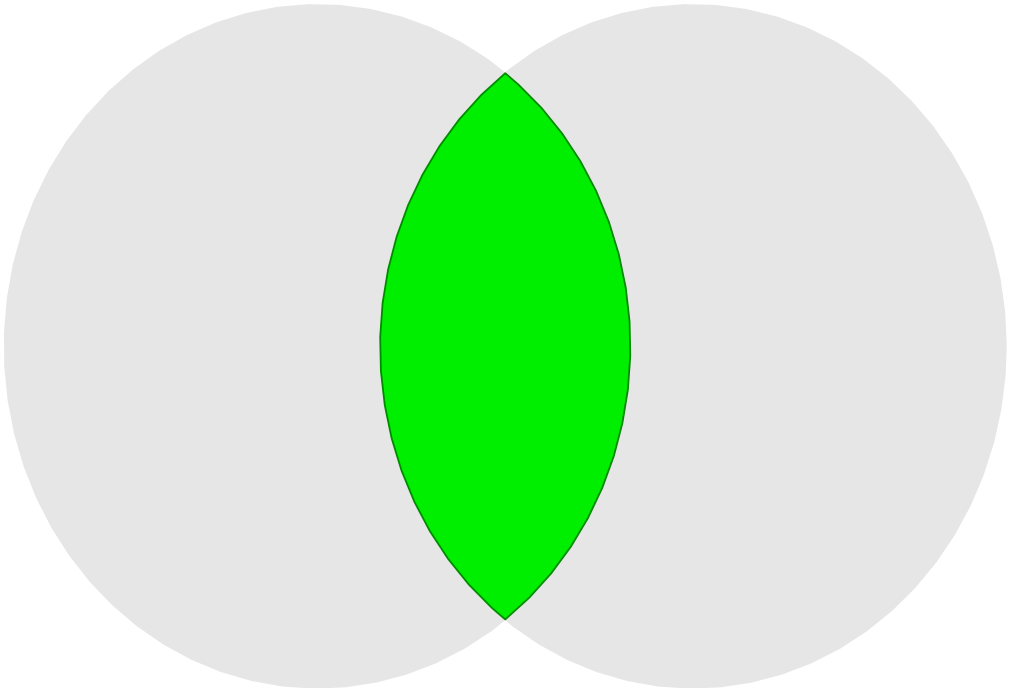
- on parcours l'arbre jusqu'au sommet qui stocke les informations à la commune
- les sommets inférieurs renseignant les suffrages
- pas besoin de fusion mais il faut parcourir l'arbre pour sélectionner les sommets satisfaisants aux conditions

  Note : mais on peut utiliser des indexes pour accélérer la recherche là aussi

Jointures entre tables

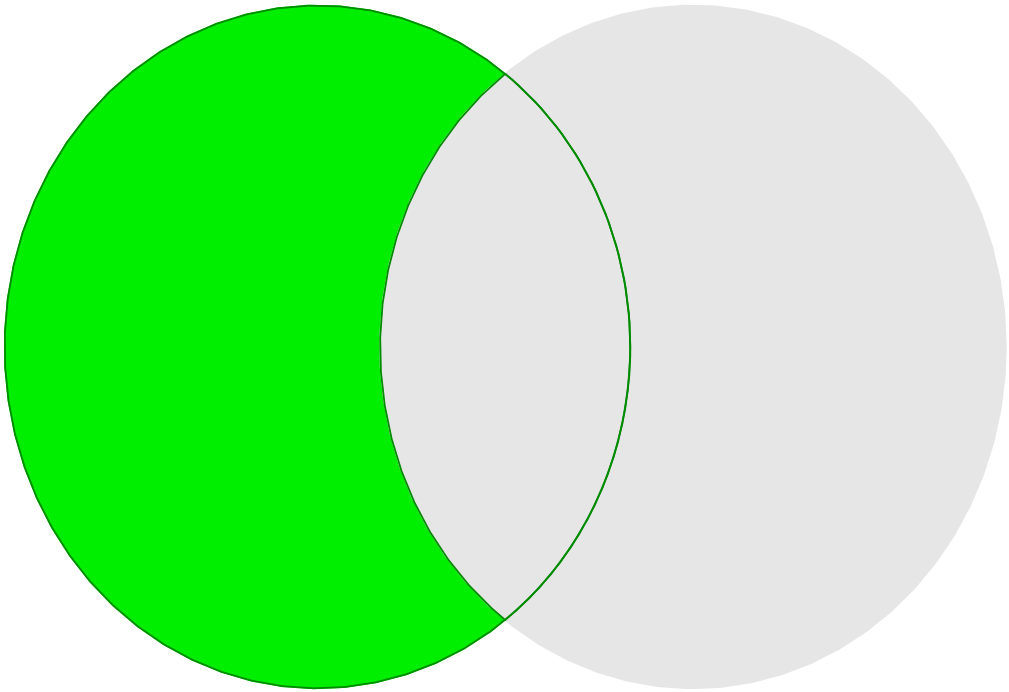
- les jointures SQL peuvent être visualisées
- au moyen de diagrammes de Venn
- utile au de–là du SQL

Intersection



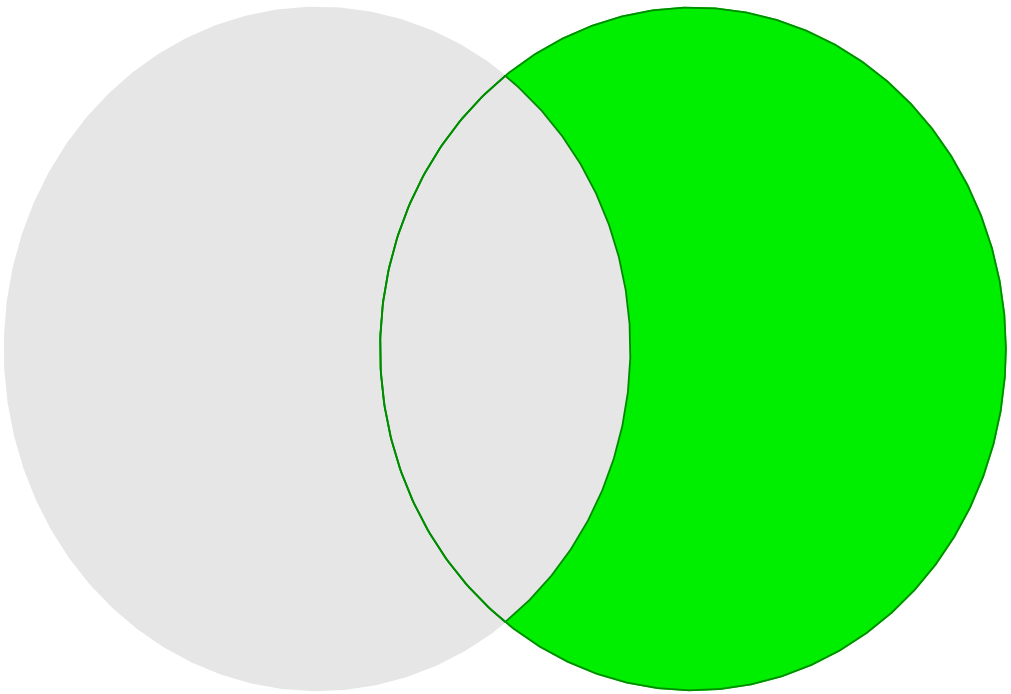
Clefs communes entre deux tables

Difference I



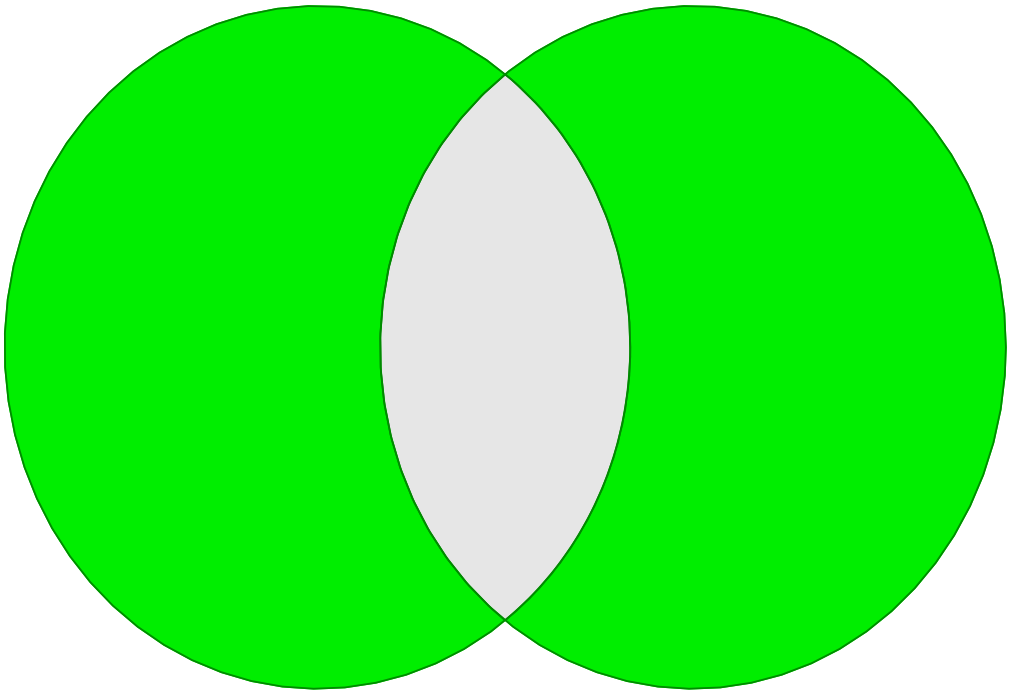
Clefs présentes dans la table A mais pas dans la table B

Difference II



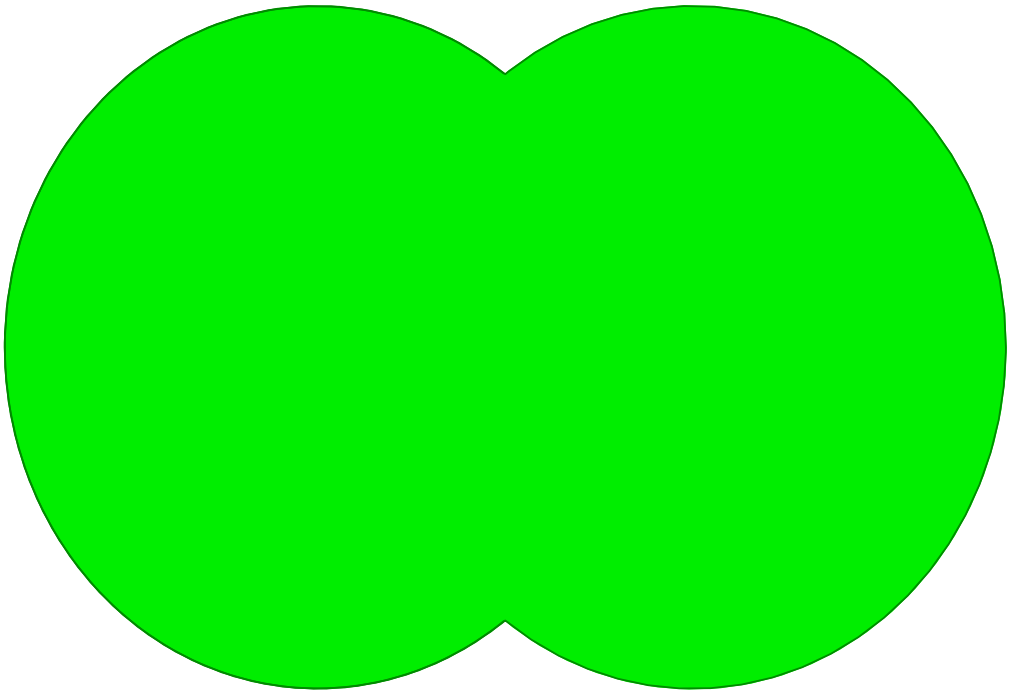
Clefs présentes dans la table B mais pas dans la table A

Difference symétrique



Clefs absentes de A ou de B

Union

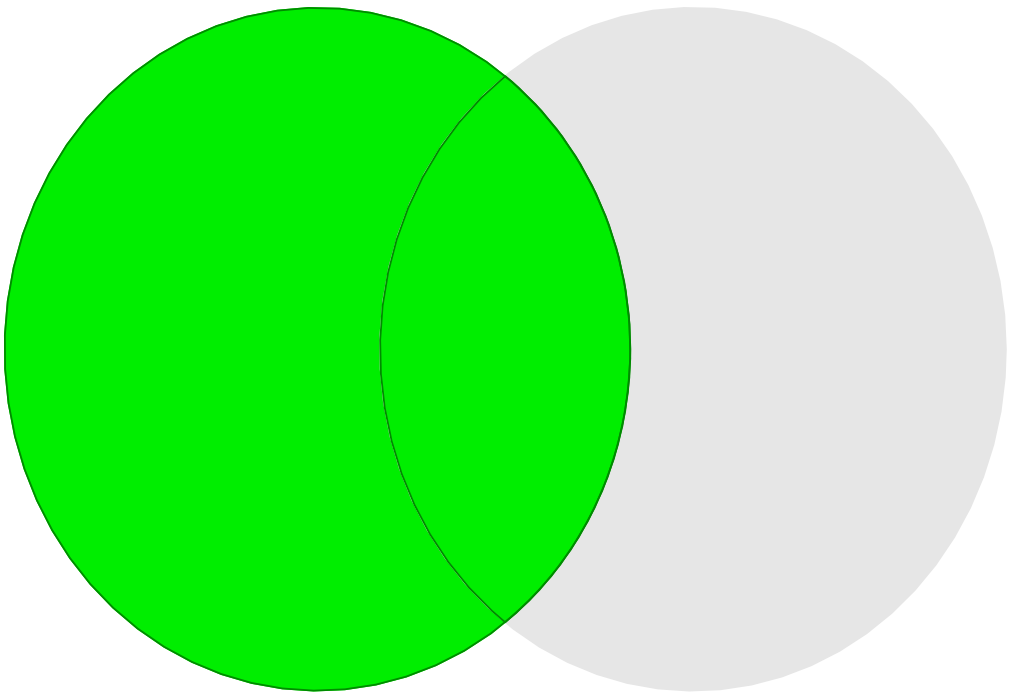


Ensembles de clefs

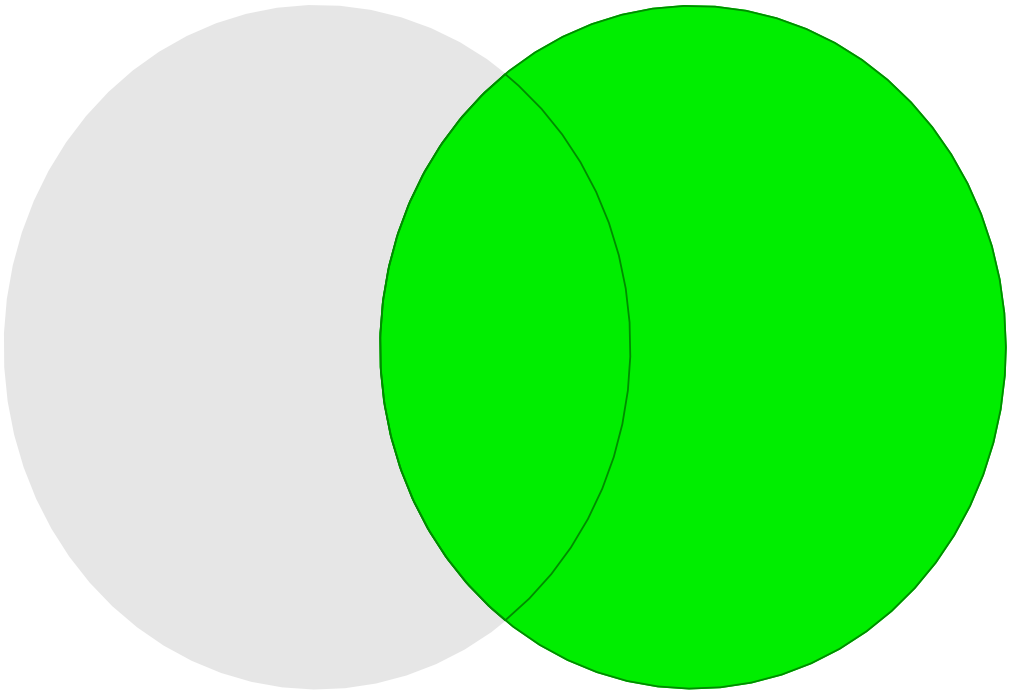
Jointures SQL

- l'intersection correspond à la jointure naturelle —`join`—
- l'union correspond à la jointure complete —`full outer join`—
- la combinaison de l'intersection et des differences permettent de définir
  - la jointure à gauche —`left join`—
  - la jointure à droite —`right join`—

Left join



Right join



Exemples avec R

fichier

`cartogaphie/postgis/scripts.R`

du dépôt en ligne

La syntaxe SQL

- synopsis d'une requête SQL

```
1  select <colnames/> -- liste des colonnes ééslectionnes
2  from <table/> -- source
3  where <cond/> -- conditions
4  ;
```

- **Exemple :** extraction à partir d'une table de données à l'IRIS

```
1  select a.* --
2  , P16_POPF/P16_POP as p16_propf -- variable écalcule
3  from iris_pop2016 a -- source + alias
4  where COM='59035' -- condition
5  ;
```

- `AS` permet d'attribuer un nom à la colonne
- `a.*` ou `a.<nom-de-la-colonne/>` signifie toutes les colonnes ou une colonne particulière de la table dont l'alias est a

Jointures

- instructions supplémentaires :
  - type de jointure
    - `join` : $(A \cap B)$ (intersection)
    - `left join` : $(A \cap B) \cup (B - A)$ (union de l'intersection de A et B et de la différence B–A)
    - `right join` : $(A \cap B) \cup (A - B)$ (union de l'intersection de A et B et et la différence B–A)
    - `full outer join` : $A \cup B$ (union)
  - les clefs d'appariement entre les tables sont stipulées par l'instruction `on` :

    a.v1= b.v1 AND a.v2=b.v2. . .
- autre jointure : produit cartesien (toutes les combinaisons de lignes)

  on ne précise pas les clefs d'appariement
- synopsis :

```
1  select <colnames/> --
2  from <table#1/> a -- sources
3  <join/> <table#2/> b
4  on a.<colname#1/> = b.<colname#2/> ...
5  where <cond/> -- conditions
6  ;
```

PostGIS

- avant de passer à la pratique, quelques remarques concernant PostGIS
- comme on va le voir, `sf` est très similaire à PostGIS

  inspiré ?

- dans la syntaxe
  - les fonctions ont généralement le même nom
  - et font presque toujours la même chose
  - on peut donc facilement passer de l'un à l'autre

  attention toutefois aux exceptions

- mais aussi dans la structure de données
  - on rajoute tout simplement une colonne de type `geometry`
  - à la table des attributs
  - qui renseigne les coordonnées géographiques de l'objet
- ce qui offre notamment la possibilité de réaliser des jointures complexes
  - combinant jointure spatiale et jointure sur les attributs
  - et en y ajoutant des filtres de sélection

Organisation des données de PostGIS et `sf`

- extrait de la couche `etab59` :

  établissements scolaires du départements du Nord

numero_	applltn	...	geometry
0593601U	Ecole élémentaire Roger Salengro	...	POINT (712072.2 7026738)
0594698L	Ecole élémentaire Salengro	...	POINT (700882.4 7059409)
0594698L	Ecole élémentaire Salengro	...	POINT (690273.4 7065401)
...	...	...	...

- dans les deux cas, une colonne contient les informations géographiques
- avec `sf`, elle s'appelle `geometry`
- avec PostGIS, on peut lui donner le nom que l'on veut

Exemples avec R

fichier

`cartogaphie/postgis/postgis.R`

du dépôt en ligne