RAPPORT

IRSN
INSTITUT
DE RADIOPROTECTION
ET DE SÛRETÉ NUCLÉAIRE

*Faire avancer la sûreté nucléaire*

# Base de données TEMPETES ET SUBMERSIONS HISTORIQUES

## Description et Guide d'Utilisation

## Rapport IRSN/2020-00858


## Pôle Santé et Environnement

Service de caractérisation des sites et des aléas naturels



MEMBRE DE
ETSON

# SOMMAIRE

## ANNEXES

# 1 INTRODUCTION

Au sein de l'IRSN, le Bureau d'expertise en hydrogéologie, sur les risques d'inondation, météorologiques et géotechniques (BEHRIG) est chargé de :

- la contribution directe aux évaluations de rapports de sûreté d'installations nucléaires de base (INB) dans les domaines de l'hydrologie, la météorologie, l'hydrogéologie, la géotechnique ;
- la réalisation d'études et recherches en support à l'expertise des dossiers de sûreté sur la caractérisation des aléas d'inondation et des aléas climatiques.

En particulier, le BEHRIG mène des travaux de collecte et d'analyse de l'information historique relative à des tempêtes et submersions sur le littoral métropolitain Atlantique, Manche et les littoraux voisins.

Dans le cadre de ces travaux, tout type de données susceptible de contenir des informations sur un événement de tempête et/ou de submersion sur le littoral français et voisin est analysé. Plusieurs ouvrages ont déjà été publiés à ce sujet, tels que Garnier et Surville (2010) [1], Aude (2006) [2], Lamb (1991) [3], Lang et Coeur (2014) [4]. On peut noter également les travaux de thèse de J.-F. Breilh (2014) [5], P. Letortu (2013) [6], S. Noel (2016) ainsi que différentes études menées par des historiens notamment E. Garnier (Université de Franche-Comté), T. Sauzeau et J. Peret (CRIHAM, Université de Poitiers) [7], par différentes institutions comme le BRGM [8, 9] ou le Shom [10] ainsi que des projets communs comme le projet VIMERS (Météo-France, Shom, CEREMA) [11]. En ligne, il existe également quelques sites web avec des informations plus ou moins détaillées sur ce type d'événements.

Lors des journées REFMAR 2016, l'intérêt d'un groupe de travail pluridisciplinaire sur le sujet des submersions et tempêtes historiques a été souligné. Depuis, ingénieurs, chercheurs, statisticiens et historiens appartenant à différents organismes (IRSN, EDF, Shom, BRGM, ARTELIA, UPLC17, CEREMA, LIENSs) se réunissent au sein du groupe de travail (GT) TEMPETES ET SUBMERSIONS HISTORIQUES dans le but de mutualiser les informations de tempêtes et submersions historiques au sein notamment d'une base de données commune. L'IRSN assure la gestion et la maintenance de la base ainsi que l'organisation et l'animation du GT.

La base de données (BD) développée par l'IRSN a été retenue par les membres du GT comme un outil de mutualisation des apports de chacun.

Dans un premier temps, les informations relatives aux événements recensés étaient regroupées au BEHRIG dans plusieurs tableaux Excel, mais plus le nombre d'événements et de sources augmentait, plus la gestion des différents tableaux devenait complexe. Ce constat a conduit à la mise en place d'une base de données relationnelle à partir des tableaux Excel.

Ainsi, la base de données TEMPETES ET SUBMERSIONS HISTORIQUES a été construite avec le logiciel PostgreSQL et est accessible via l'interface d'utilisation pgAdmin4. PostgreSQL est un logiciel libre de système de gestion de base de données relationnelle et objet (SGBDRO) qui permet également le traitement de données spatiales via l'extension PostGIS et la visualisation via un logiciel de cartographie, QGis. L'accès et la modification des données et tables se fait par le langage SQL (Structured Query Language). L'environnement R peut également être utilisé pour consulter et traiter les informations contenues dans la base de données.

Ce présent rapport présente dans un premier temps la version v3.0 de la base de données et sa structure (chapitre 2). Les chapitres 3 et 4 sont issus d'un tutoriel créé en juin 2016 et adapté aux fonctionnalités de la version v3, et

décrivent les outils informatiques et leur installation (chapitre 3) ainsi que la prise en main du langage SQL et de la base de données (chapitre 4).

# 2 LA BASE DE DONNEES

La première BD TEMPETES ET SUBMERSIONS HISTORIQUES a été mise en place en interne IRSN fin 2015. Suite aux discussions au sein du GT TEMPETES ET SUBMERSIONS HISTORIQUES, la structure de la base a été consolidée, et fin 2017 une première version est diffusée (cf. le rapport RT/PSE-ENV/2018-00066 pour la documentation associée). Au cours de l'été 2019, une nouvelle version v2.0 de la base est publiée, suite aux travaux de stage M2 de Marceau Bodin au sein de l'IRSN [12] ( documentation détaillée dans le rapport PSE-ENV/2020-00489).

Début 2020, trois étudiants de la licence universitaire professionnelle SIG proposée par l'Université de La Rochelle[1], ont été amenés à travailler sur la mise à jour structurelle de la base de données et à développer une interface SIG pour accéder aux données.

La Figure 1 présente la structure de la base de données v3.0.

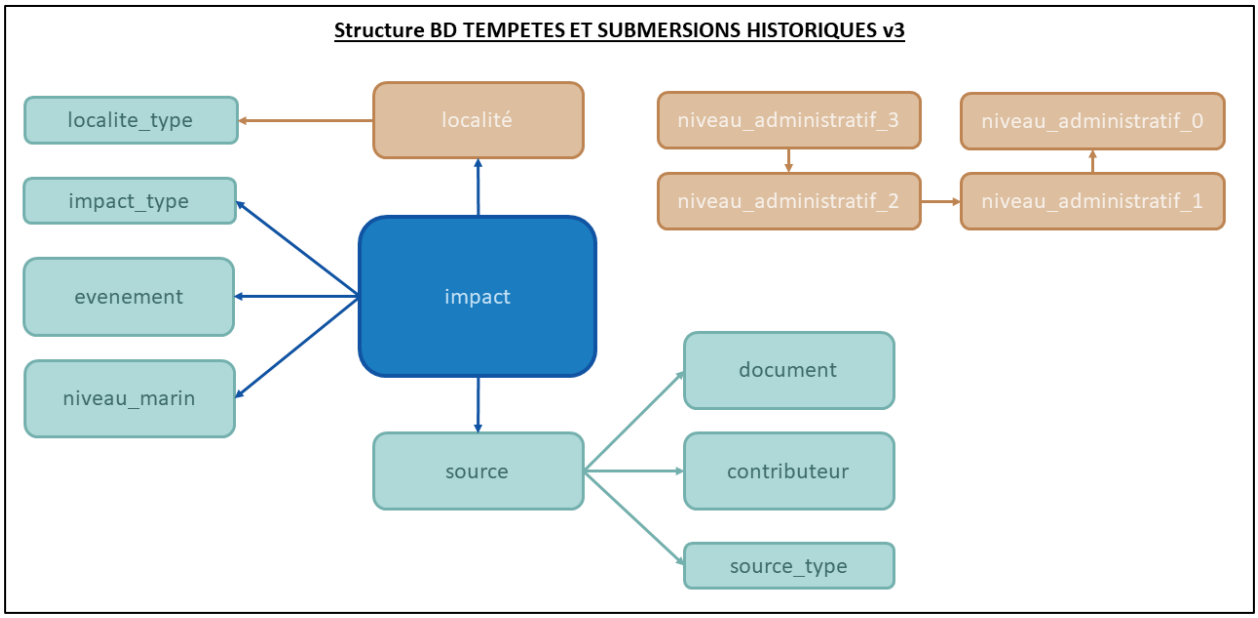



***Figure 1 : Structure de la BD version 3.***

La transmission de la base de données se fait par l'IRSN via le @gforge (*https://gforge.irsn.fr/gf/project/bdts/*). Elle reprend l'ensemble du contenu publiable, et est référencée pour une version à une date donnée. Elle contient uniquement des données publiques et est soumise à la licence suivante : https://opendatacommons.org/licenses/odbl/1.0/; une traduction française est disponible sous http://vvlibri.org/fr/licence/odbl-10/legalcode/unofficial.

Afin de pouvoir centraliser l'intégration de nouveaux événements dans la base de données, un fichier Excel est également transmis sur le @gforge. Les utilisateurs de la base de données peuvent remplir ce fichier Excel avec les informations relatives à de nouveaux événements, l'IRSN centralise les informations puis les intègre dans la base de données publique afin d'en publier une version mise à jour.

[1] https://formations.univ-larochelle.fr/lp-systemes-information-geographique

Il est à noter que la base de données publique peut être complétée/modifiée par les utilisateurs pour leurs besoins particuliers, afin de constituer une base de données spécifique. Ainsi, l'IRSN dispose d'une base de données interne (contenant des informations soumises à des accords de confidentialité) et les travaux du GT pourront éventuellement conduire à la création d'une base de données dédiée.

Dans ce rapport, les tables sont indiquées en minuscules avec la police *consolas* ayant la typologie suivante : `evenement`. Les champs des tables sont également indiqués à l'aide de la police *consolas* et marqués en italique. Les nouvelles tables sont marquées par un astérisque *. Les tables spatiales sont marquées par un ˚.

N.B. : Les noms de tables ne portent pas d'accent.

## 2.1 STRUCTURATION DES DONNEES

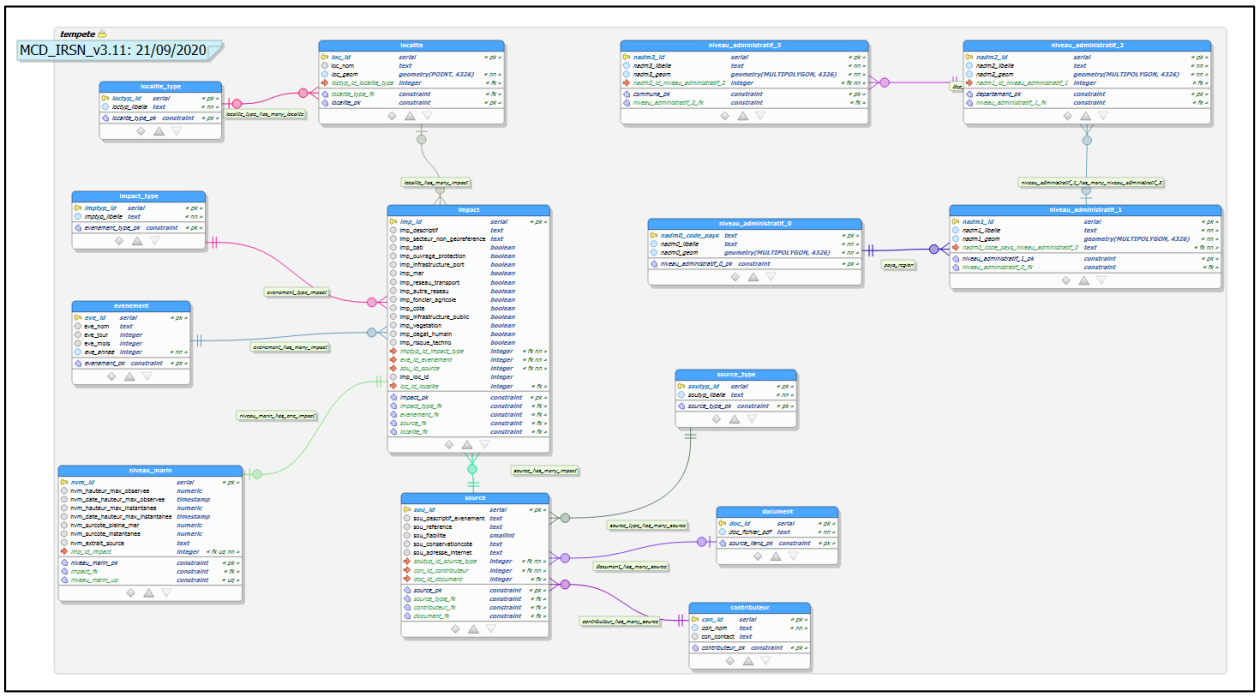



***Figure 2 : Le modèle conceptuel de données (MCD) de la BD version 3.***

Depuis sa version initiale, la structure de la BD a été simplifiée, en supprimant un grand nombre de relations entre des tables, qui étaient parfois inadaptées, et en créant des tables « adjointes » pour alléger les tables et faciliter les requêtes. Dans l'objectif de mettre en avant le côté spatial de la base de données, la table `impact` devient la table centrale de la base de données (cf. Figure 2). En effet, c'est à travers un ou plusieurs impacts qu'un événement peut être géolocalisé.

Les tables suivantes sont reliées directement à la table `impact` :

- `evenement`, contenant les informations relatives à la date d'un événement
- `niveau_marin` contenant toute information relative à un niveau marin ou une inondation
- `source` à laquelle est associée la table
    - `contributeur`
    - `source_type*`
    - `document` à laquelle est associée la table
        - `document_type*`

- localite° à laquelle sont associées les tables
    - localite_type*
    - niveau_administratif_3° à laquelle est associée la table
        - niveau_administratif_2° à laquelle est associée la table
            - niveau_administratif_1° à laquelle est associée la table
                - niveau_administratif_0).

## 2.2 MODELE PHYSIQUE DE DONNEES

### 2.2.1 *IMPACT*

Le premier objectif de cette table est de pouvoir localiser spatialement (dans un système SIG) tous les événements recensés dans la base de données. La version initiale permettait uniquement la localisation des données disponibles dans la table niveau_marin, donc seulement quand une mention de submersion était indiquée pour une localité précise.

Lors de la mise en place de cette table, au moment de la création de la version v2 de la base de données, cette table a été complétée avec 12 colonnes décrivant les dégâts à la côte lors d'un événement. Cette typologie des dégâts reprend les travaux menés lors de trois stages entre 2014 et 2016 au sein du CEREMA, dans le cadre du projet VIMERS [15].

Pour chaque type d'impact, ces champs ont été renseignés par un « 1 » (dégât recensé) ou un champ « vide » (dégât non recensé : soit il n'y a pas eu d'impact, soit l'information n'est pas connue). Il est à noter qu'une méthode de notation plus élaborée avait été mise en place au CEREMA pour analyser l'intensité des tempêtes.

***Table 1: Les champs de la table impact.***

| *Niveau Marin* | |
|---|---|
| *imp_id* | Identifiant unique pour chaque impact |
| imp_descriptif | descriptif de l'événement (issu du descriptif de la source) |
| imp_autre | nom d'un lieu/département/mer/… non-géoréférençable à partir des tables disponibles dans la base (ex : Marais de Dol) |
| imp_bati | tout type de bâti (maisons, entreprises) |
| imp_ouvrage_protection | tous les ouvrages de protection anthropiques (épi, digue,…) |
| imp_infrastructure_port | infrastructures de l'aménagement et matériel stocké (quai, mole,…) |
| imp_mer | concerne les bateaux dans les ports et au large (avarie, liaison maritime interrompue) |
| imp_reseau_transport | concerne les routes, voies ferrées |
| imp_autre_reseau | réseaux électriques, de communication, égouts |
| imp_foncier_agricole | agriculture côté terre, élevage de produits de la mer sur le littoral |
| imp_cote | érosion, franchissement |
| imp_infrastructure_public | quartier, place publique, mobilier urbain |
| imp_vegetation | arbres arrachés |
| imp_degat_humain | évacuation de population, blessés, décès |
| imp_risque_techno | risques en rapport avec des activités humaines « sensibles » |
| imp_loc_id | Est égal à *loc_id* dans la table localite<br>permet de faire le lien entre la table localité et la table impact |
| imp_imptyp_id | |
| imp_eve_id | Est égal à *eve_id* dans la table evenement<br>permet de faire le lien entre la table événement et la table impact |
| imp_sou_id | Est égal à *sou_id* dans la table source<br>permet de faire le lien entre la table source et la table impact |

### 2.2.2 *EVENEMENT*

La table **evenement** contient les informations relatives à la date d'un événement de tempête ou de submersion, cf. Table 2.

Pour chaque événement, une seule date est retenue. Arbitrairement, il s'agit de la date de la « fin » de l'événement. Par exemple, pour la tempête du 8 au 9 Janvier 1924, la date retenue sera le 9 Janvier 1924.

***Table 2: Les champs de la table evenement.***

| *Evénement* | |
|---|---|
| *eve_id* : | Un identifiant unique pour chaque date, les deux premiers chiffres correspondent au siècle |
| eve_nom | Le nom d'une tempête |
| eve_jour : | Le jour d'un événement (date fin de l'événement) |
| eve_mois | Le mois d'un événement (date fin de l'événement) |
| eve_annee : | L'année d'un événement (date fin de l'événement) |

Le champ *eve_nom* a été ajouté dans la version 3.

### 2.2.3 *NIVEAU_MARIN*

La table **niveau_marin** (cf. Table 3) est une extraction des informations qui sont stockées dans la table **source**. Alors que le descriptif d'un événement est donné de façon littéraire dans un champ unique, cette information peut être analysée puis intégrée dans la table niveau marin dans différentes colonnes. Des requêtes, faites sur le champ de la surcote de pleine mer par exemple, permettent d'extraire rapidement la surcote en une ou plusieurs localités pour un événement de submersion.

***Table 3: Les champs de la table niveau marin.***

| *Niveau Marin* | |
|---|---|
| *nvm_id* | Identifiant unique pour chaque submersion |
| nvm_hauteur_max_observee | Hauteur maximale observée |
| nvm_date_hauteur_max_observee | Date de l'observation maximale (peut être différente de la date de l'événement) |
| nvm_hauteur_max_instantanee | Hauteur du niveau marin associé à la surcote instantanée maximale |
| nvm_date_hauteur_max_instantanee | Date de la surcote instantanée maximale (peut être différente de la date de l'événement) |
| nvm_surcote_pleine_mer | Hauteur de la surcote de pleine mer |
| nvm_surcote_instantane | Hauteur de la surcote instantanée maximale |
| nvm_extrait_source | Descriptif du niveau marin |
| nvm_imp_id | Est égal à *imp_id* dans la table événement<br>permet de faire le lien entre la table impact et la table **niveau_marin** |

### 2.2.4 *SOURCE*

La table **source** (cf. Table 4) contient l'information sur un événement de tempête et/ou de submersion ainsi que les détails des sources (auteur(s), type de document, lien hypertexte …) contenant ces descriptions.

Au sein du GT TEMPETES ET SUBMERSIONS HISTORIQUES, des axes de travail sont actuellement en cours afin de mieux qualifier les types de sources et d'attribuer des indices de fiabilité à la fois aux sources utilisées et aux informations contenues [14].

*Table 4: Les champs de la table source.*

| *Source* | |
|---|---|
| *sou_id* | Identifiant unique pour chaque source |
| sou_descriptif_evenement | Le descriptif d'un événement |
| sou_fiabilite | |
| sou_con_id | Est égal à *con_id* dans la table **contributeur**<br>Permet de faire le lien entre la table contributeur et la table source |
| sou_doc_id | Est égal à *doc_id* dans la table **document**<br>Permet de faire le lien entre la table document et la table source |
| sou_reference | Référence de la source |
| sou_conservationcote | Site de Conservation et Cote du document |
| sou_adresse_internet | lien hypertexte vers la source |
| sou_soutyp_id | Est égal à *soutyp_id* dans la table **source_type**<br>permet de faire le lien entre la table événement et la table source |

### 2.2.4.1 Contributeur

Directement reliée à la table **source**, la table **contributeur** (cf. Table 5) contient les informations relatives aux contributeurs de la donnée, i.e. la personne ou l'institution ayant apporté la connaissance d'une source à l'IRSN.

*Table 5: Les champs de la table contributeur.*

| *Contributeur* | |
|---|---|
| *con_id* | Un identifiant unique pour chaque contributeur |
| con_nom | Le nom de la personne / l'institut qui a apporté la connaissance d'une source |
| con_contact | Contact mail de la personne / de l'institut qui a apporté la connaissance d'une source |

### 2.2.4.2 Document

L'objectif de la table **document** est de lister et référencer l'ensemble des documents recensés dans la base de données, cf. Table 5. Au sein de l'IRSN, un répertoire contenant les .pdf de ces documents a également été créé (transmission possible au cas par cas, notamment au sein du GT).

*Table 6: Les champs de la table document*

| *Niveau Marin* | |
|---|---|
| doc_id | Identifiant unique pour chaque document |
| doc_fichier_pdf | Nom du .pdf dans le répertoire associé |
| doc_docaut_id | Auteur(s) de l'ouvrage |

### 2.2.4.3 Source Type

Reliée à la table **source**, cette table permet la caractérisation du type de la source.

*Table 7: Les champs de la table source_type.*

| *Niveau Marin* | |
|---|---|
| soutyp_id | Identifiant unique pour chaque type de source |
| soutyp_libelle | Type de source |

Les propositions de classifications faites par Emmanuelle Athimon ont été acceptées par le GT, elles sont décrites dans la Table 8.

*Table 8: Types de source.*

| Catégorie | Définition |
|---|---|
| Source primaire **SP** | Rédigée au moment de / peu de temps après l'événement par un contemporain des faits. Le document peut être original comme une copie ou publiée par un tiers |
| **Source** secondaire **SSM** | Rédigée après l'événement et utilise des sources primaires, sur lesquelles elle produit un discours → elle contient un vécu rapporté, sans analyser l'événement |
| Littérature scientifique **LS** | Étude réalisée par des historiens, ingénieurs, chercheurs, scientifiques qui contient une analyse, une interprétation et une critique des sources historiques primaires et ou secondaires |
| Littérature technique **LT** | La littérature technique, réalisée par des bureaux d'études et/ou institutions/organisme publics **ou privés disposant d'un savoir-faire spécialisé,** est commandée par des clients. |
| Référence intraçable **RI** | Référence dont il s'avère impossible de retrouver la documentation sur laquelle se base le travail |

### 2.2.5 *LOCALITE*

Les localités françaises et européennes, touchées par des événements de tempêtes et/ou de submersions, sont recensées dans cette table. Les objets géographiques, i.e. les localités, sont de type *point* et indiquent la localisation dont la source fait mention, que ce soit un marégraphe, une commune ou un endroit spécifique dans une commune (cf 2.2.5.1).

*Table 9: Les champs de la table localité.*

| *Localité* | |
|---|---|
| loc_id | Identifiant de la localité |
| loc_nom | Nom de la localité |
| loc_geom | Information spatiale de type *point* |
| loc_loctyp_id | Est égal à *loctyp_id* dans la table **localite_type**<br>permet de faire le lien entre la table localite et la table localite_type |

#### 2.2.5.1 Localité Type

Cette table définit les types de localités afin de différencier les localités entre elles, et elle est reliée à la table **localite**. Ainsi, actuellement trois types sont présents dans cette table :

- Marégraphe : indiquant la localisation d'un marégraphe
- Ville : correspondant au centroïde d'une commune, ce type est à référencer si une localité précise n'est pas donnée dans la source
- Localité : localité exacte d'un impact suite à un événement de tempête et/ou de submersion.

*Table 10: Les champs de la table localite_type*

| *Localite_type* | |
|---|---|
| loctyp_id | Identifiant unique pour chaque type de localité |
| loctyp_libelle | Type de localité |

### 2.2.5.2 Niveau Administratifs

Dans cette version v3 de la base de données, de nouveaux découpages administratifs ont été intégrés. Ces tables spatiales, de type *polygone*, vont du découpage communal (`niveau_administratif_3`) aux limites des pays membres de l'UE (`niveau_administratif_0`). Elles sont issues des informations téléchargées sur le site internet Eurostat[2] et correspondent au découpage NUTS (Nomenclature des unités territoriales statistiques). Ainsi, toutes les communes européennes (`niveau_administratif_3`) sont représentées dans la base de données, et pas uniquement celles recensées en France par l'INSEE, comme c'était le cas dans les versions précédentes.

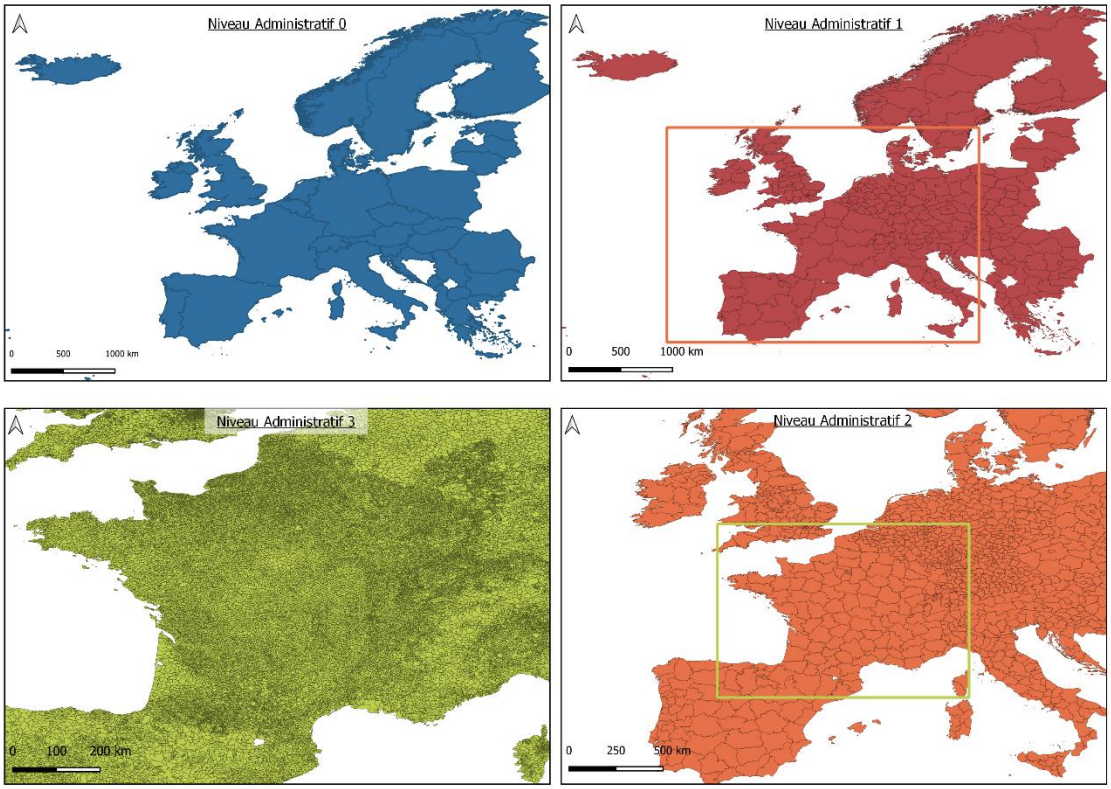



***Figure 3 : Découpage NUTS mis à disposition par Eurostat.***

Bien qu'un lien spatial existe entre les tables, une relation « dure » entre ces tables a été mise en place.

***Table 11 : Les champs de la table niveau_administratif_0***

| *Niveau Administratif 0* | |
|---|---|
| `nadm0_code_pays` | Identifiant unique pour chaque NUTS0 (pays) |
| `nadm0_libelle` | Nom du pays |
| `nadm0_geom` | Information spatiale de type *polygone* |

***Table 12 : Les champs de la table niveau_administratif_1***

| *Niveau Administratif 1* | |
|---|---|
| `nadm1_code_pays` | Identifiant unique pour chaque NUTS1 |
| `nadm1_libelle` | Nom de la NUTS1 |
| `nadm1_geom` | Information spatiale de type *polygone* |
| `nadm1_nadm0_code_pays` | Est égal à `nadm0_code_pays` dans la table `niveau_administratif_0` permet de faire le lien entre la table Niveau Administratif 0 et la table Niveau Administratif 1 |

[2] Eurostat, NUTS, https://ec.europa.eu/eurostat/fr/web/gisco/geodata/reference-data/administrative-units-statistical-units/nuts (consulté le 09/06/2020)

***Table 13 : Les champs de la table niveau_administratif_2***

| *Niveau Administratif 2* | |
|---|---|
| nadm2_code_pays | Identifiant unique pour NUTS2 |
| nadm2_libelle | Nom de la NUTS2 |
| nadm2_geom | Information spatiale de type *polygone* |
| nadm2_nadm1_id | Est égal à nadm1_code_pays dans la table **niveau_administratif_1** permet de faire le lien entre la table Niveau Administratif 1 et la table Niveau Administratif 2 |

***Table 14 : Les champs de la table niveau_administratif_3***

| *Niveau Administratif 3* | |
|---|---|
| nadm3_code_pays | Identifiant unique pour chaque NUTS3 (commune) |
| nadm3_libelle | Nom de la commune |
| nadm3_geom | Information spatiale de type *polygone* |
| nadm3_nadm2_id | Est égal à nadm2_code_pays dans la table **niveau_administratif_2** permet de faire le lien entre la table Niveau Administratif 2 et la table Niveau Administratif 3 |

### 2.2.6 *LOG*

Une table supplémentaire a été ajoutée, permettant de tracer tout ajout et / ou modification dans la base de données. Ainsi, à chaque manipulation de la base de données, une ligne est renseignée automatiquement dans cette table.

| *Log* | |
|---|---|
| log_id | Identifiant de chaque opération |
| log_nom_schema | Nom du schéma modifié |
| log_nom_table | Nom de la table sur laquelle une opération est faite |
| log_utilisateur | Nom de l'utilisateur ayant fait l'opération |
| log_date | Date et heure de l'opération |
| log_type_operation | Type d'opération |
| log_id_objet | Identifiant de l'objet modifié |

# 3 LES LOGICIELS UTILISES ET LEUR INSTALLATION

## 3.1 LES OUTILS

### 3.1.1 *L'OUTIL DE BASE DE DONNEES : POSTGRESQL*

PostgreSQL est un logiciel libre de système de gestion de bases de données relationnelle et objet (SGBDRO). PostgreSQL s'utilise via son interface pgAdmin 4, le langage de requête est le SQL (Structured Query Language), un langage informatique normalisé utilisé pour les bases de données. Cet outil est développé par une communauté mondiale de développeur et d'entreprises.

La première version a été développé dans les années 80 par Michael Stonebreaker à l'Université de Berkeley sous le nom de « Ingres », son développement est entièrement revu en 1985 et le logiciel fut renommé « Postgres » (la contraction de « Post-Ingres »). Vers le milieu des années 1990 les fonctionnalités SQL sont ajoutées à cet outil et en 1996 il fut nommé « PostgreSQL ».

### 3.1.2 *L'EXTENSION SPATIALE*

PostGIS est l'extension de PostgreSQL qui permet la manipulation d'informations géographiques qui ont une forme géométrique (point, ligne, polygone) à l'intérieur du SGBD et permet également le stockage de ce type d'information. Son nom vient de la contraction de PostgreSQL et GIS (l'acronyme anglais de SIG – Système d'information géographique).

### 3.1.3 *L'OUTIL DE CARTOGRAPHIE*

QGis est un logiciel de système d'informations géographique open-source qui permet de gérer l'extension spatiale de PostgreSQL. Comme PostgreSQL, QGis est développé par une communauté de développeur et évolue rapidement : une nouvelle version du logiciel parait environ tous les six mois.

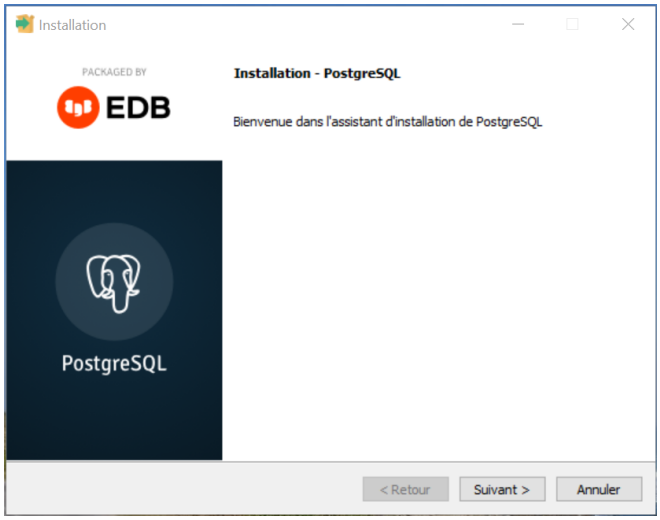
## 3.2 INSTALLATION

La base de données a été initialement développée sous PostgreSQL 9.5 et PostGIS 2.2.
La version 3 de la base de données nécessite cependant l'installation de PostgreSQL 12 et PostGIS 3.x

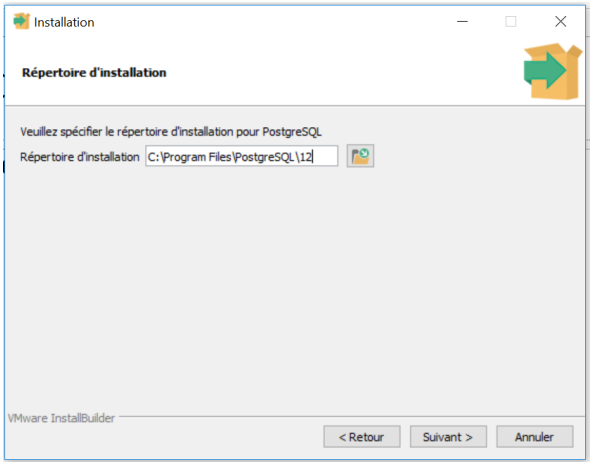
Les exécutables des installations pour le système d'exploitation WINDOWS en x32 ou x64 bit se trouvent sur le gforge https://gforge.irsn.fr/gf/project/bdts/.

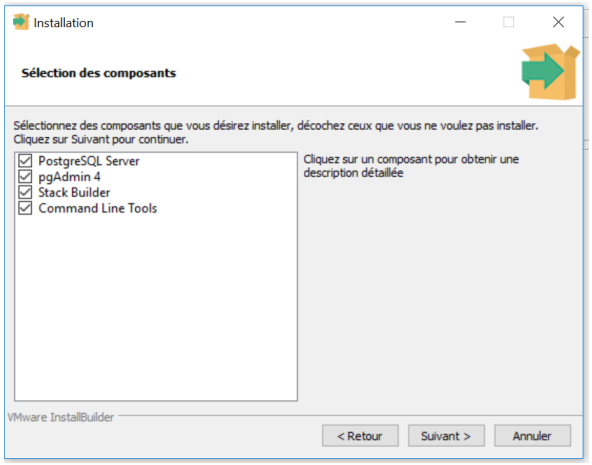
L'installation présentée ci-dessous est faite pour les exécutables en x64-bit.

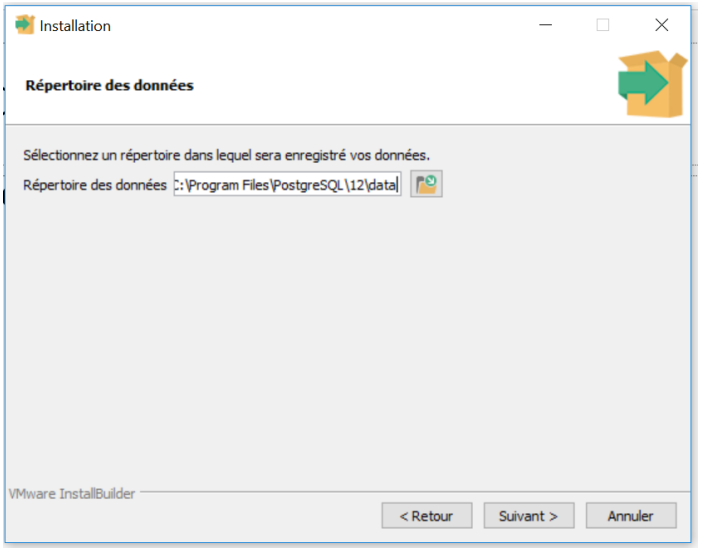
### 3.2.1 *POSTGRESQL 12*



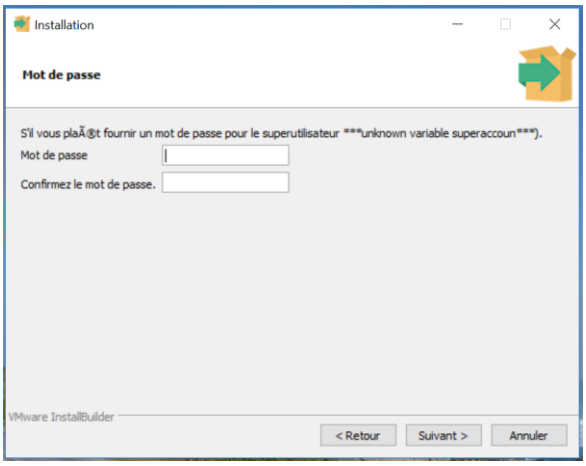

Cliquez sur suivant et laissez la sélection de l'emplacement des dossiers par défaut.

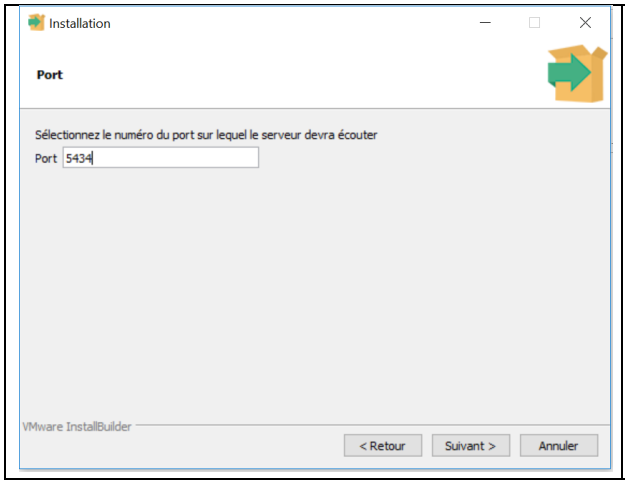


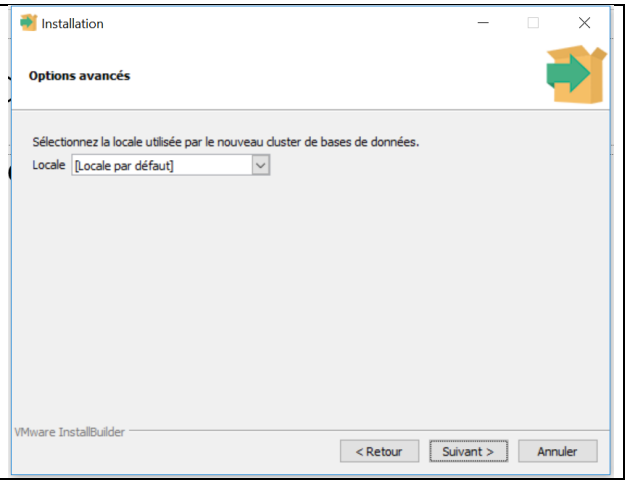



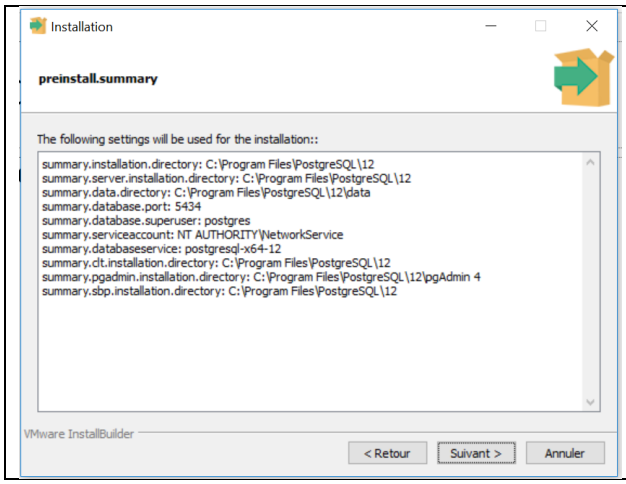

Saisir un mot de passe du super-administrateur postgres sur le serveur local. De manière générale le mot de passe configuré est **postgres** - tout attaché et en minuscule - comme le nom de l'administrateur. Pour une base de données distante, il est conseillé de choisir un mot de passe plus sûr.

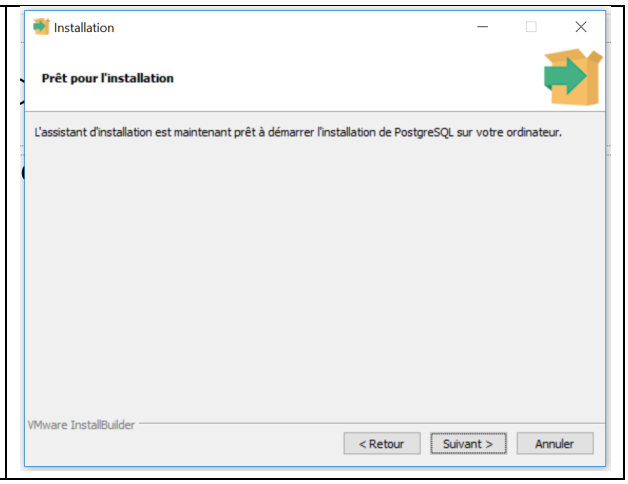


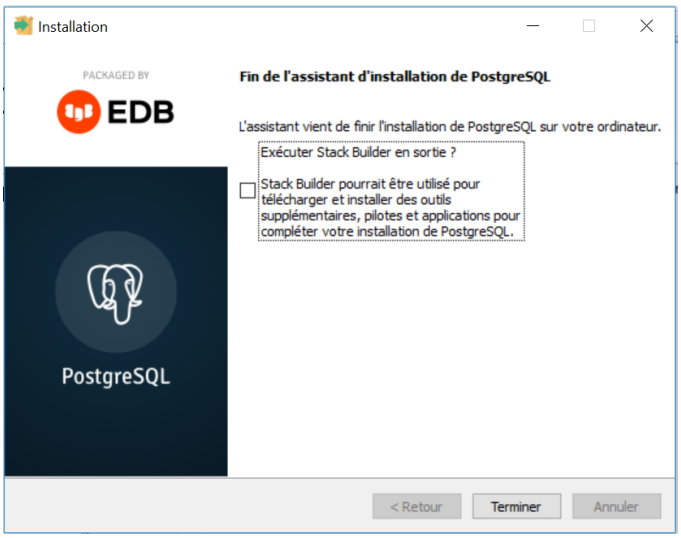




Laissez le numéro du port par défaut **5432** , dans le cas ou une autre version de PostgreSQL est déjà installée, changez le numéro de port. Gardez, le **Locale par défaut** pour le cluster des bases de données.



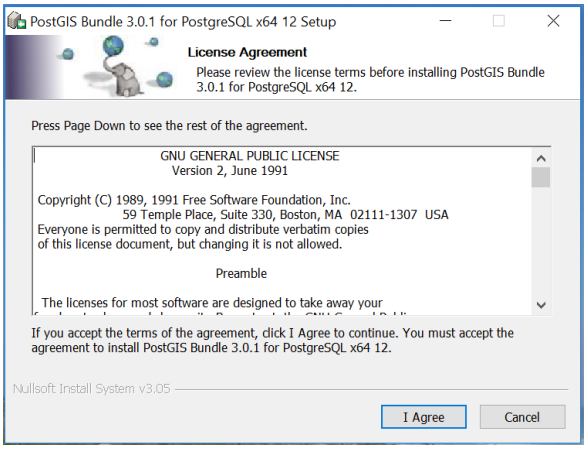




Cliquer sur Terminer, mais décochez la case qui permet l'exécution du Stack Builder, l'installation va se terminer.

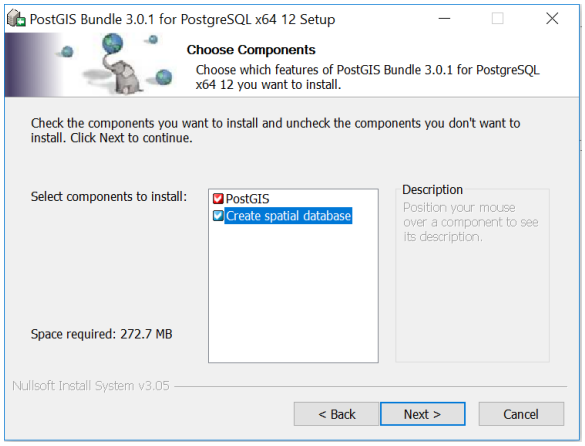



## 3.2.2 *POSTGIS BUNDLE 3*

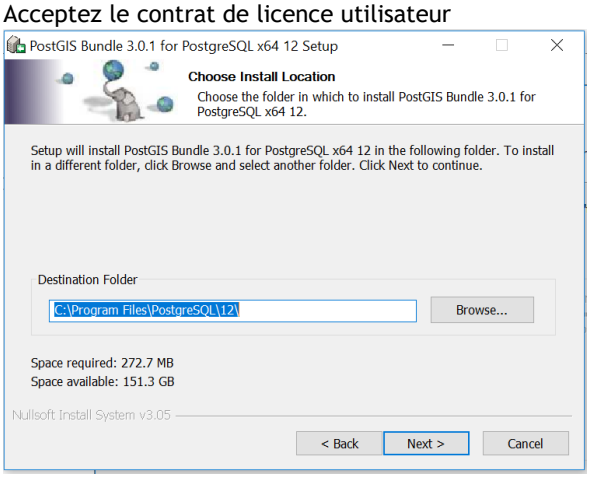
Lancez l'éxécutables de PostGIS 3.x.



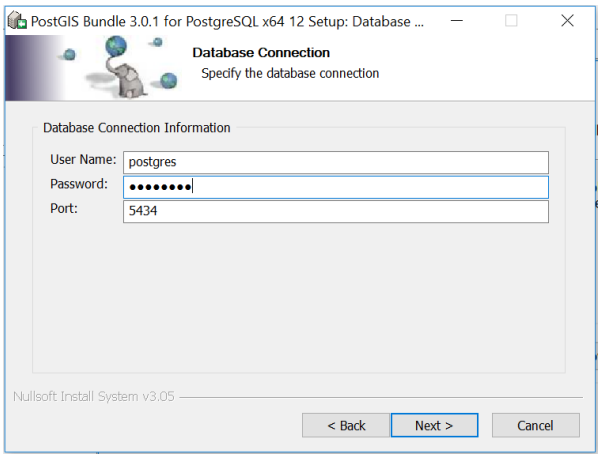

Acceptez le contrat de licence utilisateur




Cocher la case « **Create spatial database** »




Laisser le dossier d'installation des fichiers par défaut (il devrait automatiquement sélectionner le fichier PostgreSQL 12).

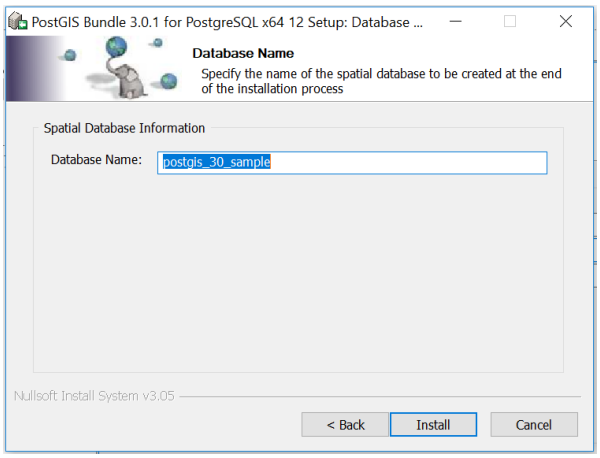



Se connecter en tant que super administrateur

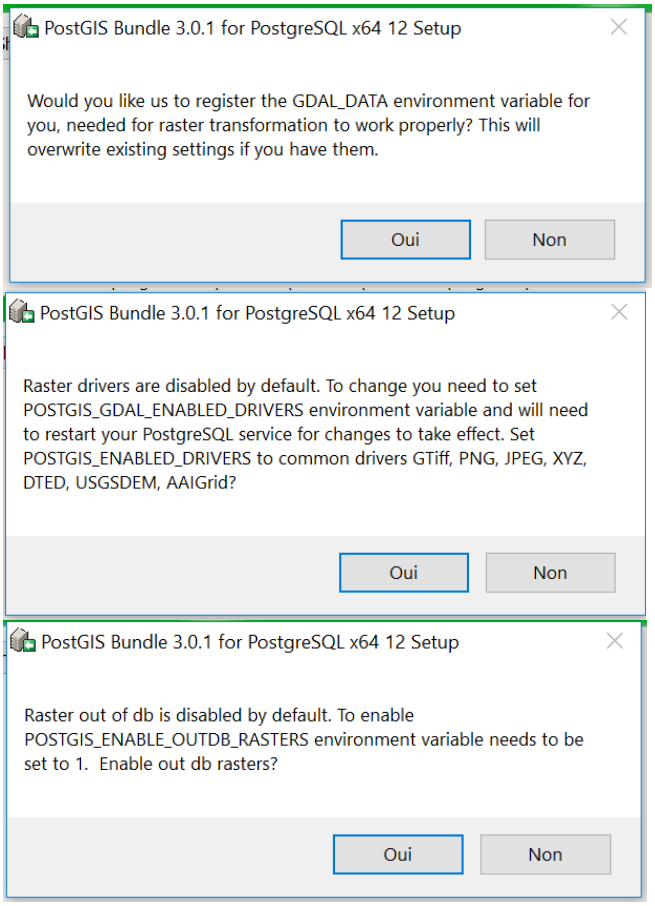
super administrateur : **postgres**

mot de passe : **postgres**

à la base de données locale sur le port **5434**

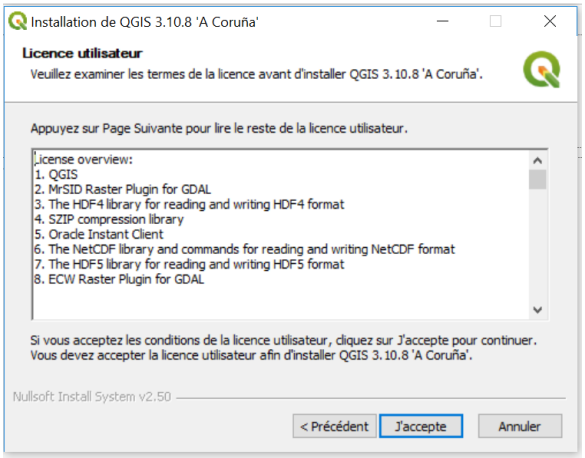


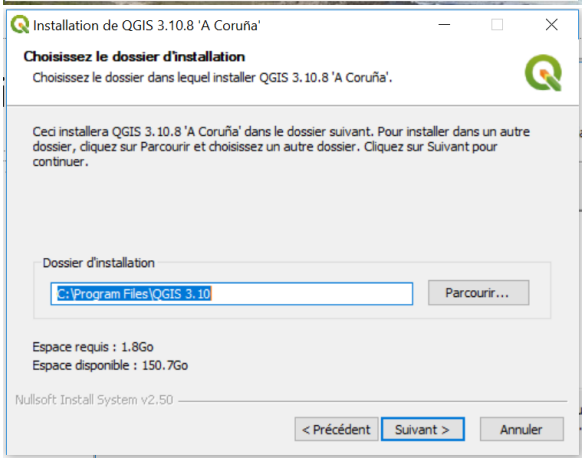

Laisser par défaut le nom de la base de données spatiale, qui sera créée à la fin de l'installation.

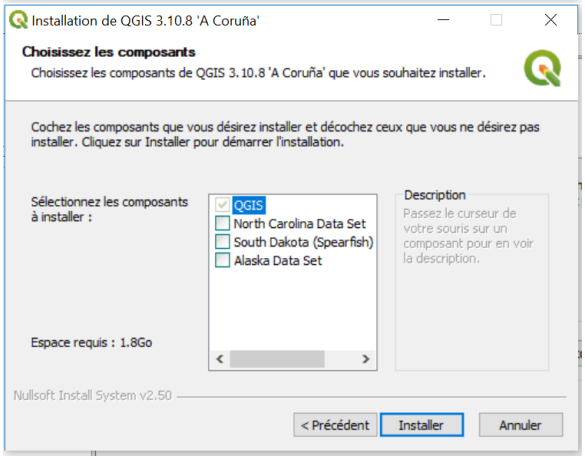


### 3.2.3 *QGIS 3.X*

Une version plus récente de QGis peut également être utilisée, la procédure d'installation sera sensiblement la même.

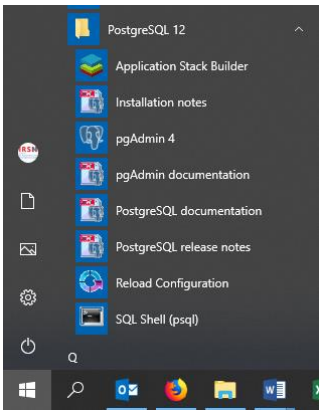



# 4 UTILISATION

## 4.1 PGADMIN 4

### 4.1.1 *OUVRIR PGADMIN 4*




Démarrer → PostgreSQL 12 → pgAdmin 4

**Documentation** : contient les outils d'aide de PostgreSQL.

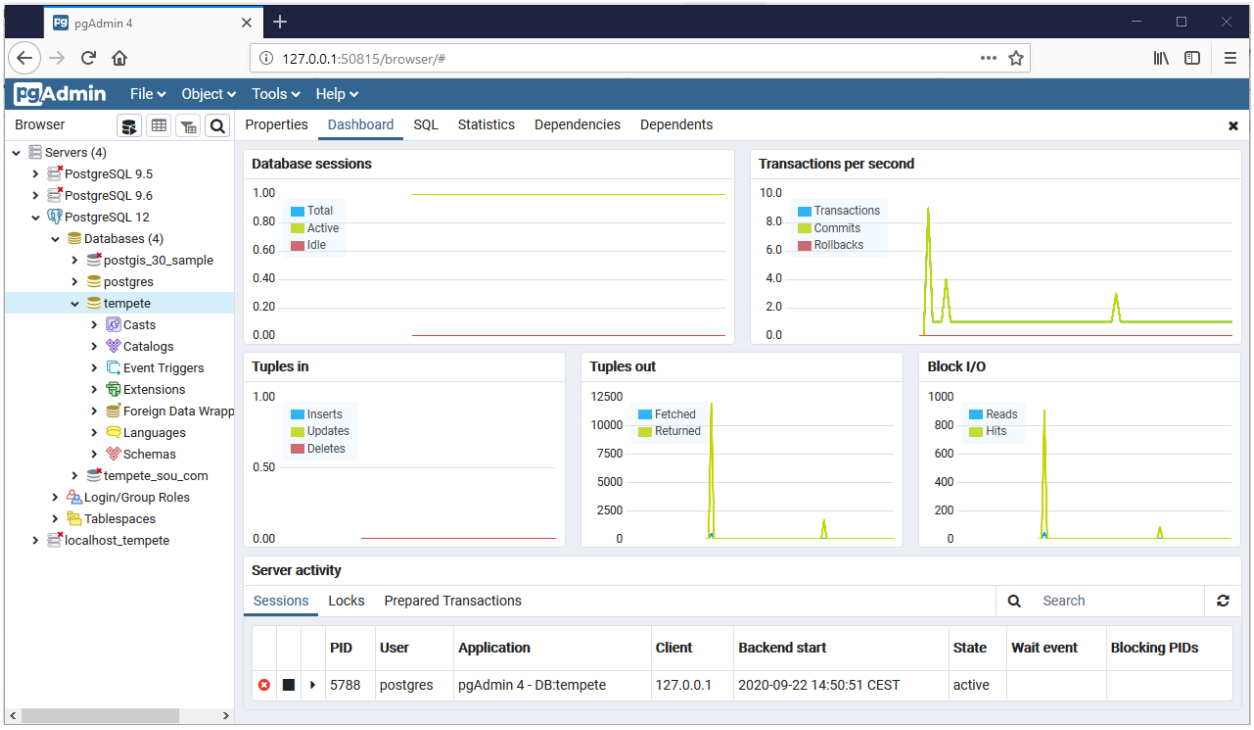
**Application Stack Builder** : permet de charger des extensions

**Reload configuration** : permet de réinitialiser PostgreSQL si des modifications de configurations ont été faites.

**SQL Shell** : utilisation de PostgreSQL en mode invite de commande, via le langage psql.

### 4.1.2 *L'INTERFACE PGADMIN 4*

Contrairement à la version précédente, pgAdmin 4 s'ouvre dans un explorateur internet.




La connexion à une base de données se fait en cliquant sur un des serveurs.

La base disponible par défaut est une base locale, c'est-à-dire que toutes les informations stockées dans cette base sont uniquement accessibles par l'ordinateur sur lequel est installé PostgreSQL ; aucune connexion à distance n'est possible.

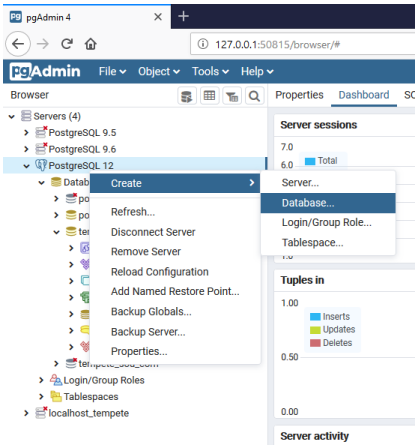
Pour pouvoir travailler sur des bases de données partagées, la connexion à un serveur distant est nécessaire.

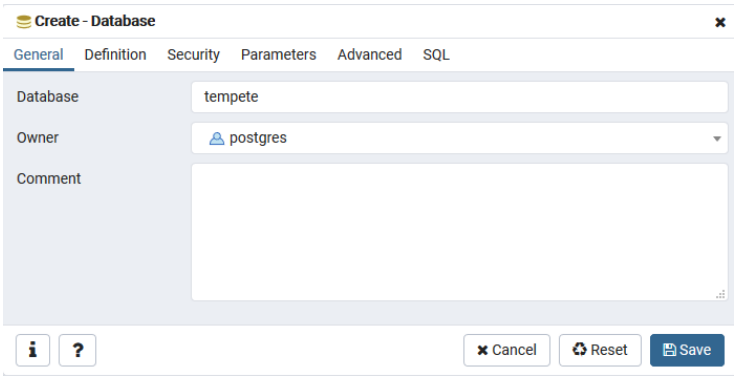
## 4.2 IMPORTER LA BASE DE DONNEES

En amont d'utilisation de la base de données, il faudra décider si la base doit être installée sur le serveur local i.e. le serveur créé lors de l'installation de PostgreSQL, ou si la base doit être installée sur un serveur partagé. Pour installer la base sur un serveur, il faut se connecter au server souhaité à l'aide de pgAdminIII.

L'exemple ci-dessous décrit l'import de la base de données sur le server local (localhost), à partir d'un fichier *xxx.backup*.

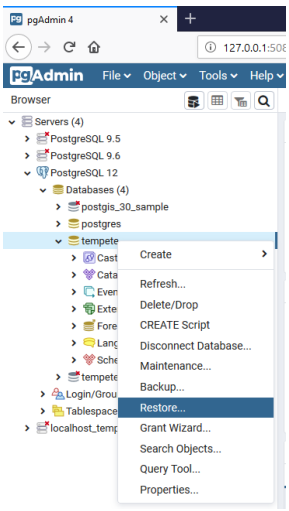
Ajouter une nouvelle base de données et la nommer **`tempete`**.

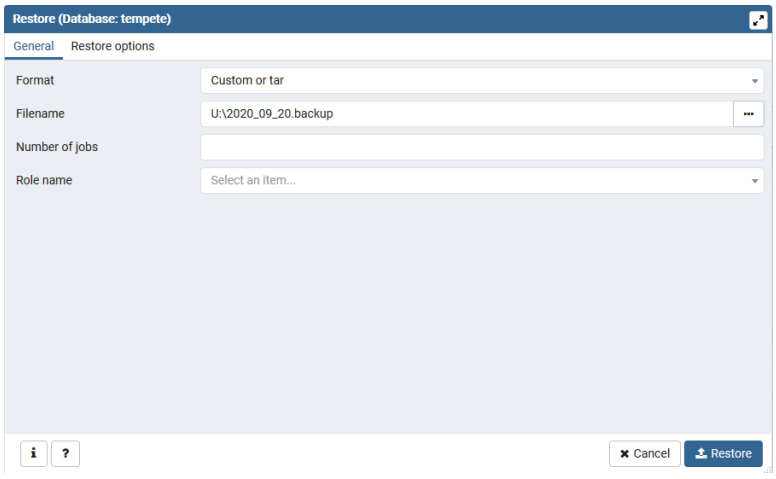



### 4.2.1 *IMPORT VIA UN FICHIER* .BACKUP

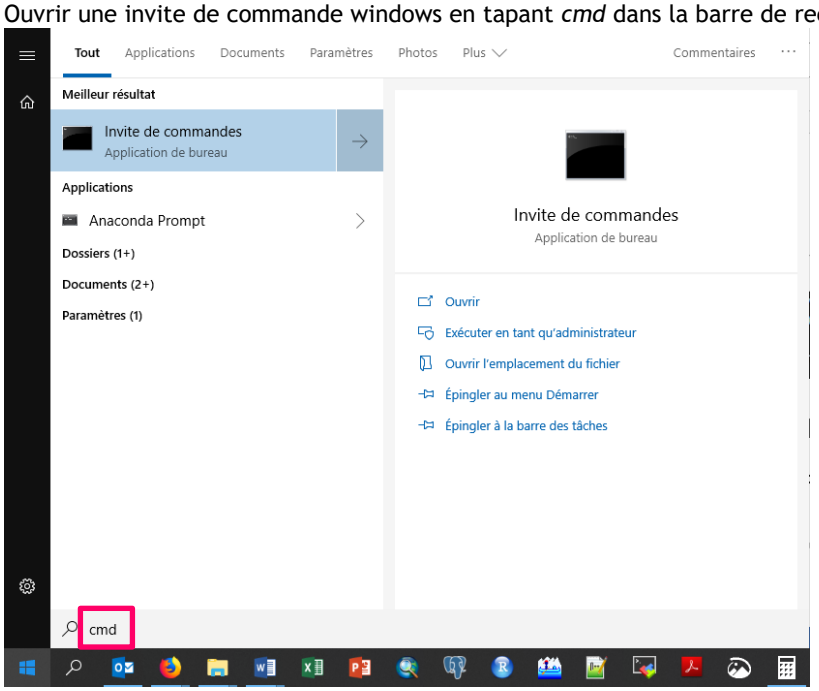
Clic droit sur cette nouvelle base → ***restore*** et charger le fichier *.backup* récupéré sur le gforge ou via clé USB.



### 4.2.2 *IMPORT VIA UN FICHIER .SQL*

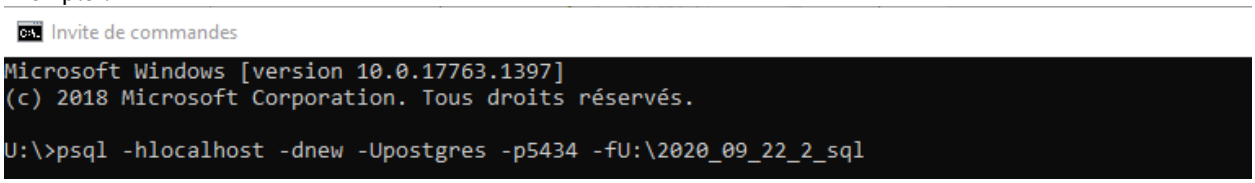
Ouvrir une invite de commande windows en tapant *cmd* dans la barre de recherche windows.




Taper la commande suivante :

```
psql –hnomserver –pnumeroport –Unomutilisateur -dnomBD –fcheminfichier
```

Exemple :

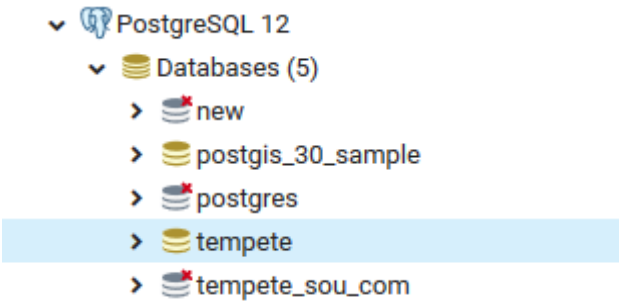



## 4.3 CONNEXION A LA BASE DE DONNEES

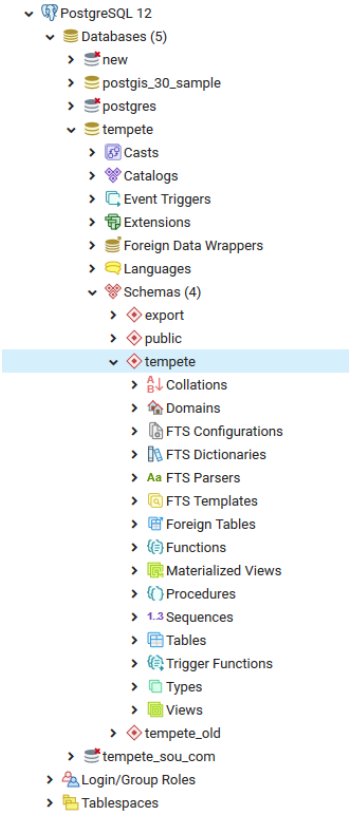
Selon le serveur sur lequel la base a été installée, l'accès à cette base va différer. La partie 4.3.1 permet de voir comment accéder à la base si celle-ci est installée sur un serveur local, la partie 4.3.2 décrit l'accès à la base si elle a été installée sur un serveur partagé.

### 4.3.1 *CONNEXION SI LA BASE EST INSTALLEE SUR UN SERVEUR EN LOCAL*

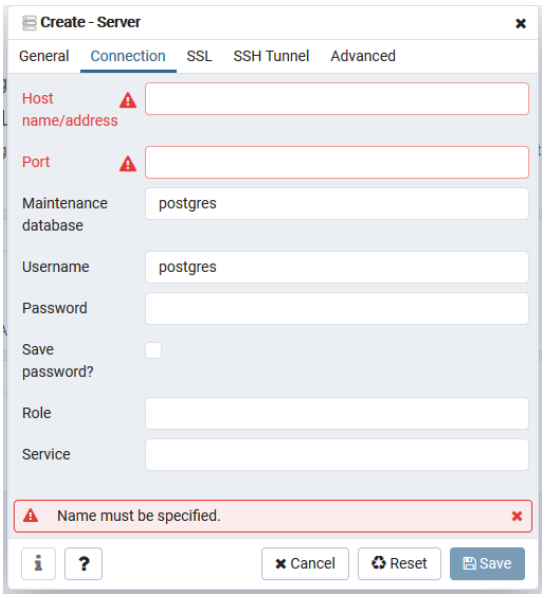
Dans un premier temps il faut se connecter au serveur local.




Dérouler le répertoire **Bases de données** et se connecter à la base souhaitée, puis sélectionner le **schéma** souhaité afin de pouvoir accéder aux **tables** de cette base de données.



### 4.3.2 *CONNEXION SI LA BASE EST INSTALLEE SUR UN SERVEUR SPECIFIQUE*

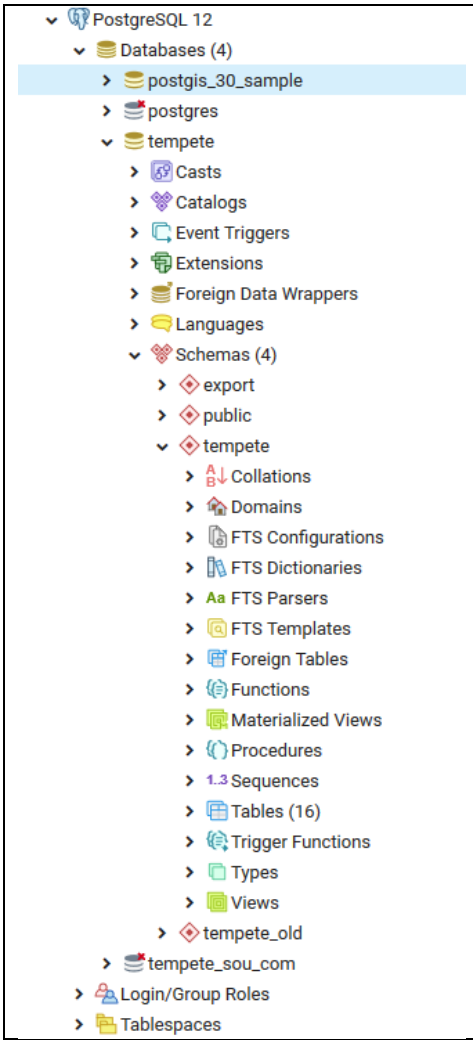



**Nom :** Nom que vous voulez donner à la BD dans l'arborescence
**Host name /adresse :** adresse IP ou nom du domaine du serveur
**Port:** numéro de port sur lequel est lancé Postgres
**Correspondant au serveur**

**Maintance database** : laisser par défaut

**Username :**
**Password**

## 4.4 CONTENU DE LA BASE DE DONNEES

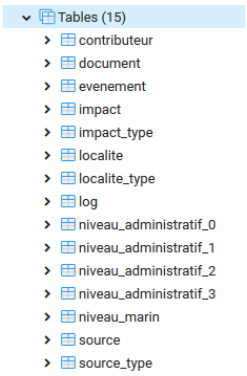



**Les branches les plus importantes sont décrites ci-dessous :**

- postgis_30_sample – installé par défaut
- postgres - installé par défaut
- tempête
  - Catalogues :
    - Contient les fonctions SQL et les fonctions de PostgreSQL
  - Extension :
    - Contient les extensions pour cette base. `plpsql` est installé par défaut ; la base TEMPETES contient les extensions `postgis` et `postgis_topology`
  - Schémas :
    - Espace de noms différents mais avec un stockage commun. On y retrouve les Tables avec nos données, les fonctions spatiales etc.

- Rôles de connexion
  - Comptes pour accéder à la base de données avec différents droits (modification, lecture uniquement…)

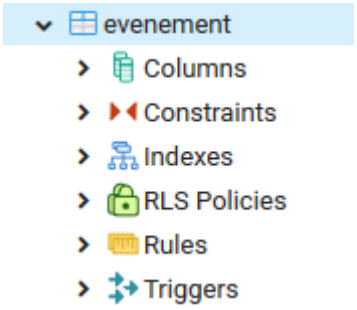
## 4.4.1 *NAVIGATION DANS LA BASE ET LES TABLES*




Dérouler l'onglet Tables pour voir le contenu :

On y retrouve les 14 tables du MCD ainsi que la table log.

En double-cliquant sur une table ou en cliquant sur le › à côté du nom de la table, un menu va se dérouler.




Columns : contient les noms des champs et le nombre de colonnes dans la table

Constraints : Clé primaire et clé(s) étrangère(s)

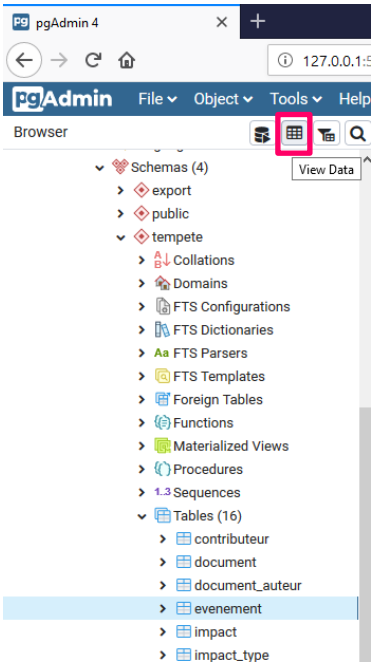
Indexes

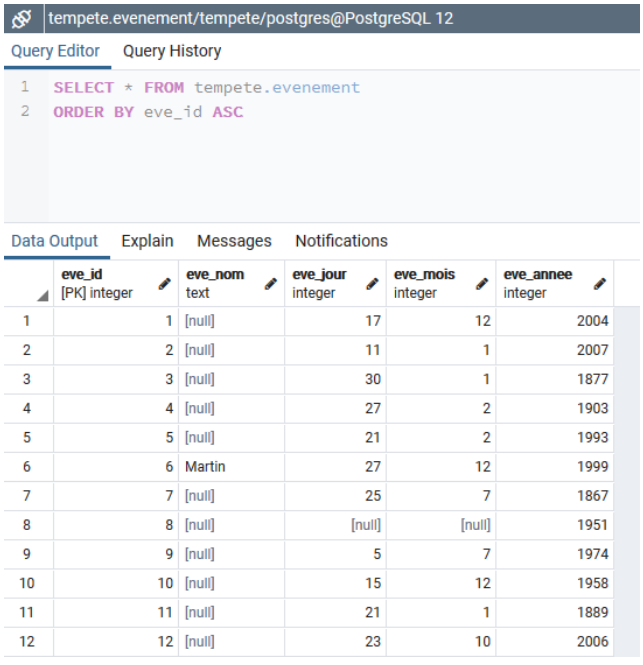
RLS Policies

Rules

Triggers : "Déclencheur", qui exécute une fonction lorsqu'un événement survient

## 4.4.2 *VISUALISER UNE TABLE*



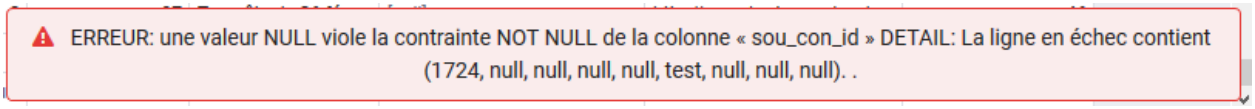

| | eve_id [PK] integer | eve_nom text | eve_jour integer | eve_mois integer | eve_annee integer |
|---|---|---|---|---|---|
| 1 | 1 | [null] | 17 | 12 | 2004 |
| 2 | 2 | [null] | 11 | 1 | 2007 |
| 3 | 3 | [null] | 30 | 1 | 1877 |
| 4 | 4 | [null] | 27 | 2 | 1903 |
| 5 | 5 | [null] | 21 | 2 | 1993 |
| 6 | 6 | Martin | 27 | 12 | 1999 |
| 7 | 7 | [null] | 25 | 7 | 1867 |
| 8 | 8 | [null] | [null] | [null] | 1951 |
| 9 | 9 | [null] | 5 | 7 | 1974 |
| 10 | 10 | [null] | 15 | 12 | 1958 |
| 11 | 11 | [null] | 21 | 1 | 1889 |
| 12 | 12 | [null] | 23 | 10 | 2006 |

## 4.4.3 *LES TRIGGERS*

A l'intérieur de cette base de données il y a également des «triggers », c'est-à-dire de des « déclencheurs », sur toutes les tables.

Lorsqu'un nouvel enregistrement est rempli ou une modification de champs est faite l'action suivante va avoir lieu : Cette fonction trigger permet de renseigner la table `log,` à l'intérieur de laquelle toute modification de la base de données est tracée. Le schéma, le nom de la table, le nom d'utilisateur, la date, le type d'opération et l'identifiant de l'objet sont enregistrés.

Dans la BD existent des triggers complémentaires, qui empêchent l'enregistrement d'une saisie, si certains champs ne sont pas remplis. Par exemple, si la référence d'une source n'est pas saisie, un message d'erreur va apparaitre.




## 4.5 REQUETES

Le langage SQL est un langage spécifique aux bases de données. Il permet d'interroger la base, de modifier des champs et des lignes, mais également de modifier la structure de celle-ci.

### 4.5.1 *REQUETE DE SELECTION EN LANGAGE SQL*

En cliquant sur ce symbole, un éditeur de requête va s'ouvrir, dans lequel nous allons écrire notre requête.

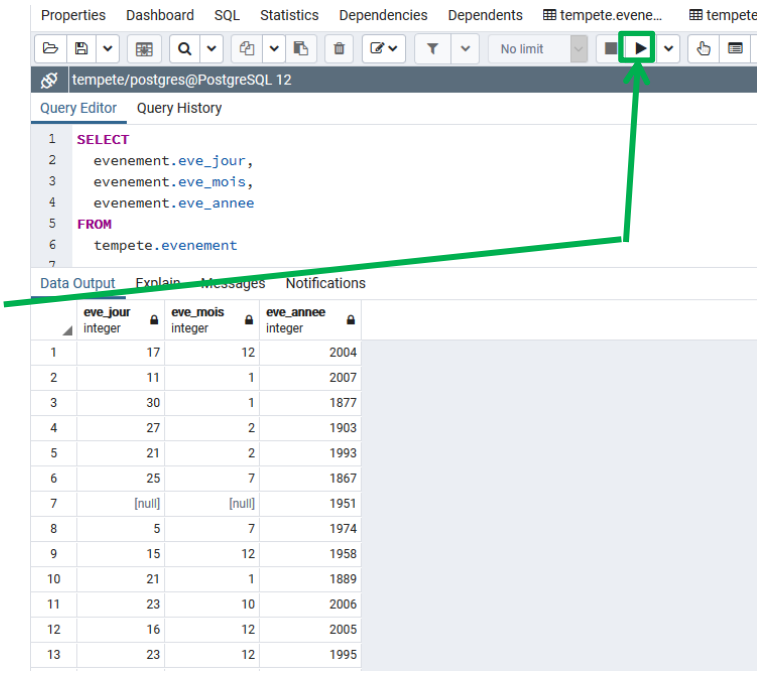
La structure de base d'une requête

| | |
|---|---|
| `SELECT` | Choix du champ dans la table |
| `FROM` | Choix de la table dans le schéma |

Nous savons que les dates de tempêtes sont stockées dans la table `evenement` et dans le champ `eve_date`. La table est stockée dans le schéma `tempete`.

La requête sera alors :

```
SELECT
 evenement.eve_jour,
 evenement.eve_mois,
 evenement.eve_annee

FROM
 tempete.evenement
```





permet d'exécuter la requête

Pour trier les données par ordre chronologique il suffit d'ajouter:

```
SELECT
 evenemenet.eve_date
FROM
 tempete.evenement
ORDER BY eve_annee, eve_mois, eve_jour ASC
```

Quelle est la date de la tempête Lothar ?

Pour répondre à cette question, nous avons besoin d'effectuer une requête sur le champ `eve_nom` et chercher l'enregistrement correspondant :

- La commande **WHERE** permet d'imposer un critère à la requête et
- L'opérateur **LIKE** permet de faire une requête sur les caractères.

Effectuez la requête suivante :

```
SELECT
  evenement.eve_jour,
  evenement.eve_mois,
  evenement.eve_annee
FROM
  tempete.evenement
WHERE
  eve_nom LIKE 'Lothar%'
```


L'opérateur **LIKE** permet de trouver des correspondances dans une table selon un motif (une chaîne de caractère) défini dans la requête.

Ici nous souhaitons voir tous les enregistrements dont le nom stocké dans le champ `eve_id` correspond à Lothar.

% remplace n'importe quelle chaîne de caractère ; il peut être placé au début, eu milieu ou, comme dans cet exemple, à la fin d'une chaîne.

## 4.5.2 *REQUETE PAR CRITERE NUMERIQUE*

Dans la partie précédente nous avons vu l'opérateur `LIKE`, qui permet de faire une requête sur les caractères.

Dans PostgreSQL il existe également des opérateurs de comparaison, comme dans R ou le langage python.

| ***Opérateur*** | ***Description*** |
|---|---|
| < | inférieur à |
| > | supérieur à |
| <= | inférieur ou égal à |
| >= | supérieur ou égal à |
| = | égal à |
| <> ou != | différent de |

1. Quelles sont les dates de tempête du 21ème siècle ?
2. Combien d'événements ont eu lieu au 17ème siècle ?

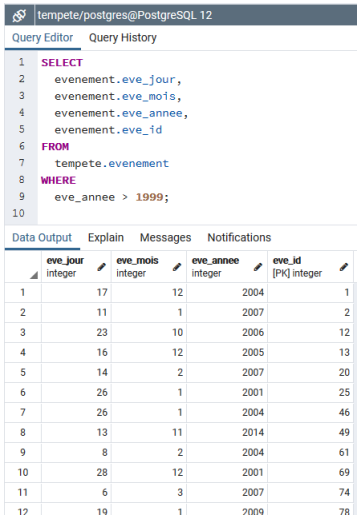
1. Quelles sont les dates de tempête du 21ème siècle ?

Pour cela, nous avons besoin de renseigner plusieurs choses dans la requête :

- Sélectionner les enregistrements ayant une année supérieure à l'année 1999
- imposer un critère à la requête en ajoutant la commande `WHERE`.

Effectuez la requête suivante :

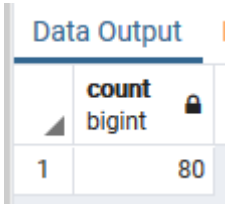
```
SELECT
  evenement.eve_jour,
  evenement.eve_mois,
  evenement.eve_annee,
  evenement.eve_id
FROM
  tempete.evenement
WHERE
  eve_annee > 1999;
```




2. Combien d'événements ont eu lieu au 17ème siècle ?

Pour compter le nombre d'événement d'un siècle nous allons modifier la requête en y ajoutant le paramètre `COUNT()` :

```
SELECT
  COUNT(evenement.eve_annee)
FROM
  tempete.evenement
WHERE
  (eve_annee < 1800 and eve_annee > 1699);
```



**Quelques requêtes supplémentaires (les solutions sont données en annexe).**

1. Compter les submersions qui ont touché Brest.
2. Quelles sont les villes touchées par une tempête – ex. 9 Janvier 1924, avec les infos complémentaires
3. Quelles sont les dates de tempêtes ayant causées une submersion supérieure à 50 cm à La Rochelle ?
4. Trouver les sources pour un événement
5. Compter les sources pour un événement

## 4.5.3 *CREATION DE TABLES A PARTIR D'UNE REQUETE*

### 4.5.3.1 Création de Table

Actuellement toutes les données utilisées ou créées à l'aide de requêtes sont stockées sur le serveur en local ou à distance de PostgreSQL, mais ne sont pas écrites en dur sur l'ordinateur ou le réseau de travail. Pour pouvoir extraire ces données, les résultats d'une requête seront inscrits dans une table, qui sera exportée par la suite.

Afin de ne pas modifier la structure du schéma `tempete`, dans lequel les informations de la base de données sont stockées, il faut créer une nouvelle table, qui sera exportée dans un schéma dédié, appelé `export`. Pour cela, en amont de la requête, il faut ajouter la ligne suivante :

```
CREATE TABLE export.descriptif AS
```

Dans une table `export.descriptif` sélectionner toutes les tempêtes par date ainsi que le descriptif.
Compter le nombre de sources pour les événements du 19ème siècle.

La première ligne de la requête suivante permet de créer la table appelée `descriptif` dans le schéma `export`.

```
CREATE TABLE export.descriptif AS

SELECT
  evenement.eve_jour,
  evenement.eve_mois,
  evenement.eve_annee,
  source.sou_descriptif_evenement
FROM
  tempete.evenement,
  tempete.source,
  tempete.impact
WHERE
  (evenement.eve_id = impact.imp_eve_id AND
  source.sou_id = impact.imp_sou_id)
group by
  (evenement.eve_jour,
  evenement.eve_mois,
  evenement.eve_annee,
   sou_descriptif_evenement)
ORDER BY
  eve_annee ASC
```

Une fois la requête exécutée, le message suivant apparaitra :

Data Output | Explain | Messages | Notifications

```
SELECT 1658

Query returned successfully in 163 msec.
```

La table apparait bien dans le schéma `export`.

En essayant de ré-exécuter la requête, un message d'erreur apparaitra :

Data Output | Explain | Messages | Notifications

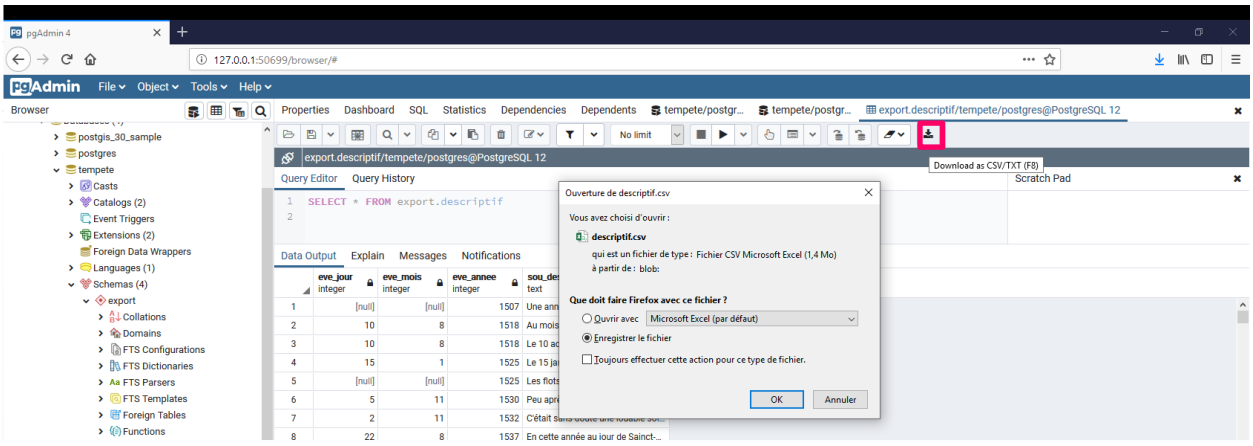
```
ERROR: ERREUR:  la relation « descriptif » existe déjà


SQL state: 42P07
```

En effet, PostgreSQL ne peut pas recréer la table, même si la requête a été modifiée. Il faut alors soit renommer la table, soit ajouter une ligne en amont de la requête, qui va permettre de supprimer l'ancienne table pour réécrire la nouvelle.

```
DROP TABLE IF EXISTS export.descriptif;
CREATE TABLE export.descriptif AS
```

### 4.5.3.2 Exporter une table

Le symbole  permet d'enregistrer une table en .csv ou en .txt.



# 4.6 VOLET SPATIAL – QGIS

## 4.6.1 *QUELQUES NOTIONS*

PostGIS est l'extension spatiale du SGBDRO PostgreSQL, permettant de créer et de gérer des bases de données qui peuvent être utilisées par un système d'information géographique – SIG. L'abréviation PostGIS vient de la contraction de Postgres et GIS (Geographic Information System).

Les formes géométriques supportées par un SIG sont :
- Point
- Linestring
- Polygone
- Multipoint
- Multilinestring
- Multipolygone
- Geometrycollection

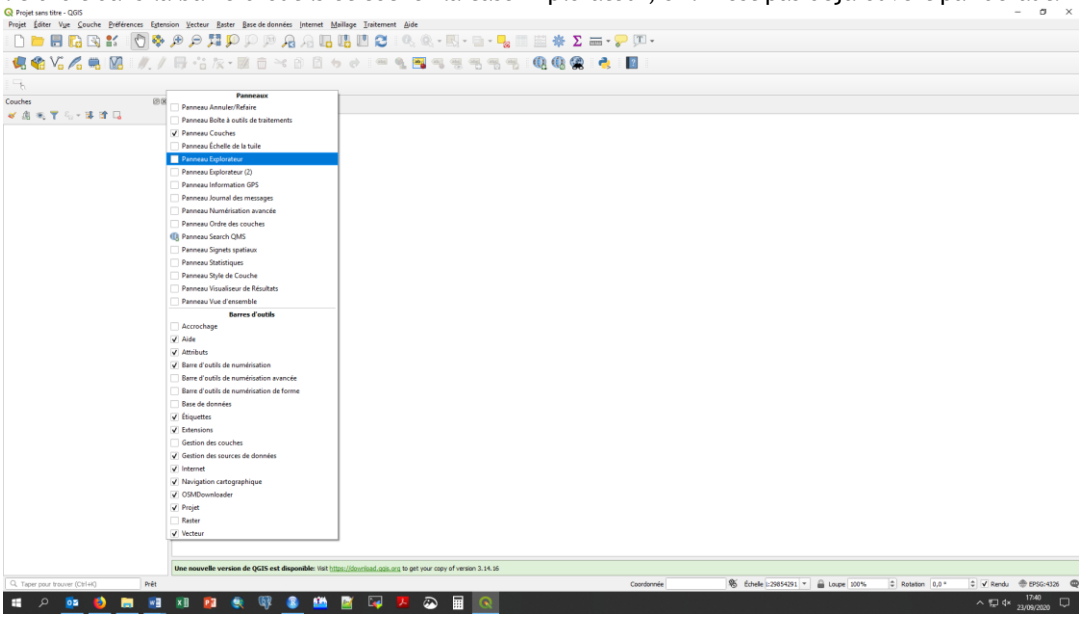
Les données, que l'on souhaite ajouter dans une base spatiale, peuvent être ajoutées sous la forme de WKT (Well-Known Text), un format de données textuel défini par l'Open Geospatial Consortium http://www.opengeospatial.org/. Le logiciel de base de données transformera ces informations en données géographiques.

L'information géométrique est stockée dans les champs ayant le suffixe *_geom* dans les tables spatiales.

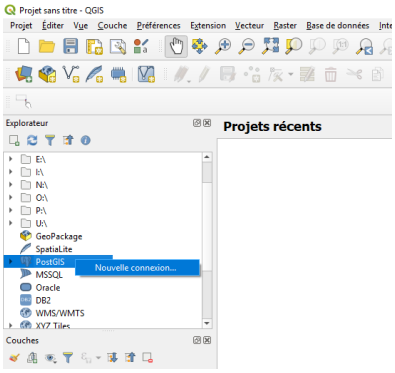
## 4.6.2 *CONNEXION DE LA BASE A QGIS*

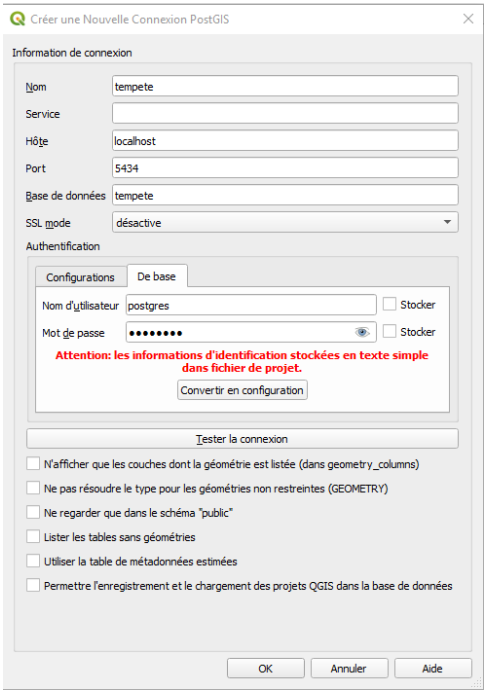
Pour se connecter ouvrez QGis Desktop et se connecter à la base de données.

Clic droit dans la barre d'outils et cocher la case Explorateur, s'il n'est pas déjà ouvert par défaut.




Pour se connecter à une base de données, cliquez sur PostGIS dans l'explorateur et ajouter une nouvelle connexion.




Nom : le nom que vous souhaitez donner à cette connexion (peut différer du nom de la base)

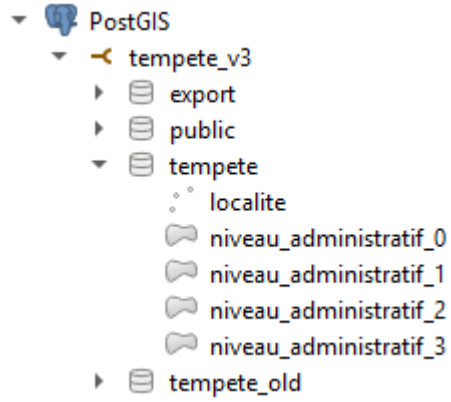
Service : laisser vide

Hôte : le nom de l'hôte sur lequel la base est installée

Port : le port sur lequel est installée la base

Base de Données : le nom de la base

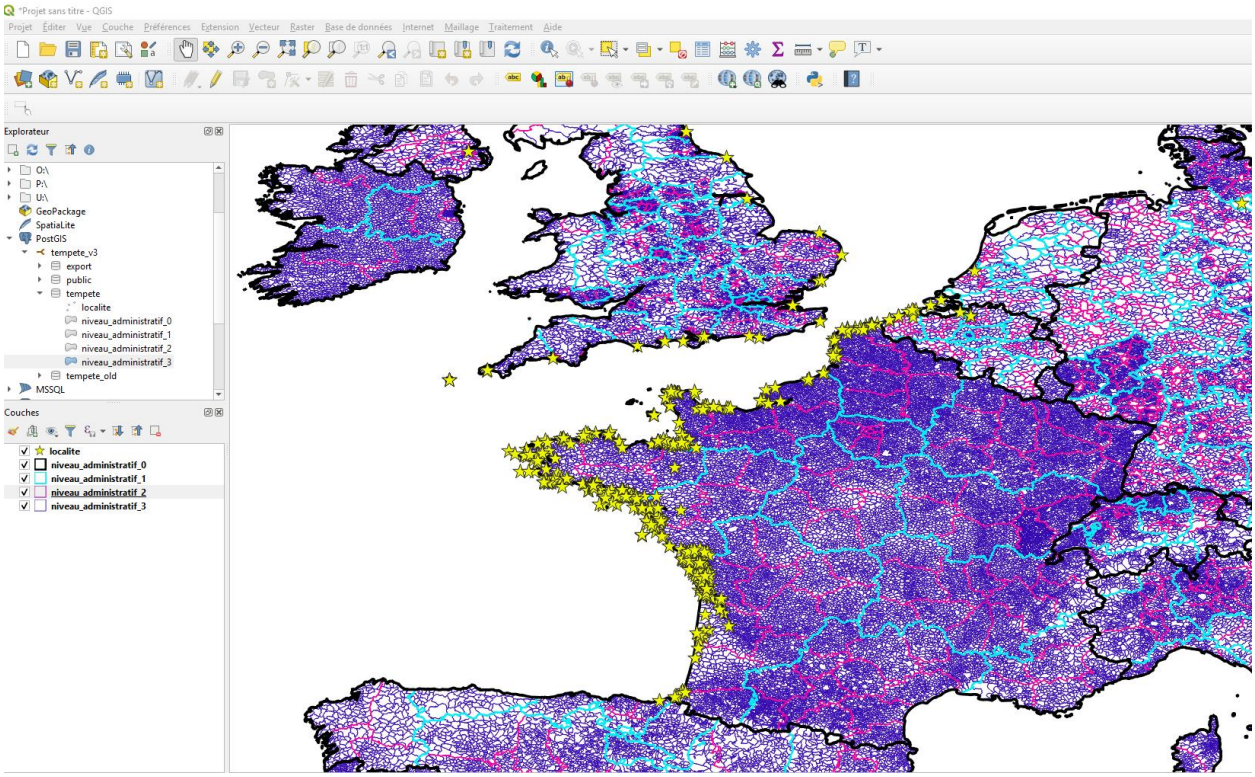
En déroulant dans l'explorateur l'onglet connexion PostGIS, la base apparait.




*Click droit → Propriétés → Style* sur une couche permet de modifier le symbole / couleur de remplissage.

*Click droit → Ouvrir la table des attributs* sur une couche permet de voir les éléments dans la table.

La table des attributs montre les mêmes données que la table dans pgAdmin 4, sans la colonne `geom`, la géographie étant affiché dans le logiciel SIG.




## 4.6.3 *LES CALCULS A PARTIR DE COUCHES SPATIALES*

Les couches dans la base de données tempête sont référencées en WGS84, un système de coordonnées exprimé en degrés[3]. Nous allons re-projeter les données dans le système de projection français officiel, Lambert93[4], un système où les coordonnées sont exprimées en mètre. Chaque référence spatiale a un identifiant, appelée **SRID** (Spatial Reference Identifier). Le SRID de la projection utilisée pour la base de données WGS84 est le 4326. Le **SRID** correspondant au système Lambert93 est **2154**.

| Projeter la couche ville en Lambert 93 |
|---|

La requête ci-dessous va

- créer une nouvelle table appelée `loc_l93`
- exporter dans le schéma `export`
- projeter en Lambert93 en reprenant les champs `loc_id, loc_nom et loc_loctyp_id,` de la table `localite` dans le schéma `tempete`
- transformer le champ `geom` en Lambert 93 (2154) et en le nommant loc_`geom2`

[3] http://geodesie.ign.fr/index.php?page=glossaire#gwgs84 (07/11/2017)
[4] http://geodesie.ign.fr/index.php?page=rgf93 (07/11/2017)

```
CREATE TABLE export.loc_l93 AS

SELECT
row_number () over () AS gid,  -- creer nouvel identifiant
loc_id,
loc_nom,
loc_loctyp_id,
st_transform(loc_geom, 2154) AS loc_geom2 -- transformer la projection en L93
FROM tempete.localite;
```

La commande `AS gid`, ou `AS geom2`, permet de donner un nom à un champ, lors de sa création.

*Attention : il se peut que lorsque vous copiez le code, les commentaires (données en marron ci-dessus) ne s'affichent pas en tant que commentaires dans la fenêtre de requête SQL. Le plus simple c'est de supprimer ces commentaires.*

Calculer la distance entre des points (villes) et classer les distances par ordre croissant.

Il existe des fonctions géométriques, qui s'intègrent dans la requête SQL :

La fonction que nous allons utiliser est la fonction `st_distance`

```
SELECT
  entourage.loc_nom,
  st_distance(centre.loc_geom2, entourage.loc_geom2) AS distance
FROM
  export.loc_l93 AS centre,
  export.loc_l93 AS entourage
WHERE
  centre.loc_nom LIKE 'LA ROCHELLE' AND
  centre.loc_nom!=entourage.loc_nom
ORDER BY distance ASC
```

Quelques explications pour comprendre la requête :

- Lors de la sélection de couche, nous allons sélectionner deux fois la table loc_93 en lui donnant deux noms différents: `centre` et `entourage`

  ```
  FROM
    export.loc_l93 AS centre,
    export.loc_l93 AS entourage
  ```

- Dans la commande `WHERE` nous allons préciser que nous souhaitons uniquement l'enregistrement où le nom de la ville est `'LA ROCHELLE'` et que la couche centre n'est pas égale à entourage - ce qui nous permet d'exclure le point `'LA ROCHELLE'` lors du calcul de distance (en effet la distance entre `'LA ROCHELLE'` et `'LA ROCHELLE'` est de 0).
-

```
WHERE
  centre.loc_nom LIKE 'LA ROCHELLE' AND
  centre.loc_nom!=entourage.loc_nom
```

Et voici le résultat :

*Displaying records 1 - 10*

| loc_nom | distance |
|---|---|
| LA ROCHELLE – Port Neuf | 846.3726 |
| LA ROCHELLE - Le Mail | 1048.7012 |
| LA ROCHELLE - Les Minimes | 1801.9837 |
| LA ROCHELLE - VIEUX PORT MAREGAPHE 1 | 1886.9461 |
| LA ROCHELLE - VIEUX PORT MAREGAPHE 2 | 1908.1414 |
| LA ROCHELLE - Vieux Port | 1909.7080 |
| LA ROCHELLE - VIEUX PORT MAREGAPHE 3 | 1914.3747 |
| LA ROCHELLE - Chef de Baie | 2363.1096 |
| LA ROCHELLE - LA PALLICE MAREGAPHE | 3544.3235 |
| NIEUL-SUR-MER | 5239.7003 |

L'unité de mesure de la projection Lambert93 étant en mètre, la distance calculée entre les localités et La Rochelle est également donnée en mètres. Si l'on souhaite avoir la distance en kilomètres, il suffit de diviser le résultat par 1000 comme indiquée dans la ligne ci-dessous :

```
st_distance(centre.geom2, entourage.geom2)/1000 AS distance
```

## 1. Quels sont les marégraphes dans un rayon de 200 km autour d'une ville choisie ?

Cette requête va se faire en 2 temps.

Dans un premier temps nous allons délimiter un cercle, avec un rayon de 200 km autour de la ville choisie, (souvent appeler « buffer » ou « zone tampon » dans les SIG) puis réaliser une intersection pour répondre à notre question initiale.

Nous voulons obtenir les marégraphes dans un rayon de 200 km autour de La Rochelle :
La commande PostGIS est `st_buffer()`

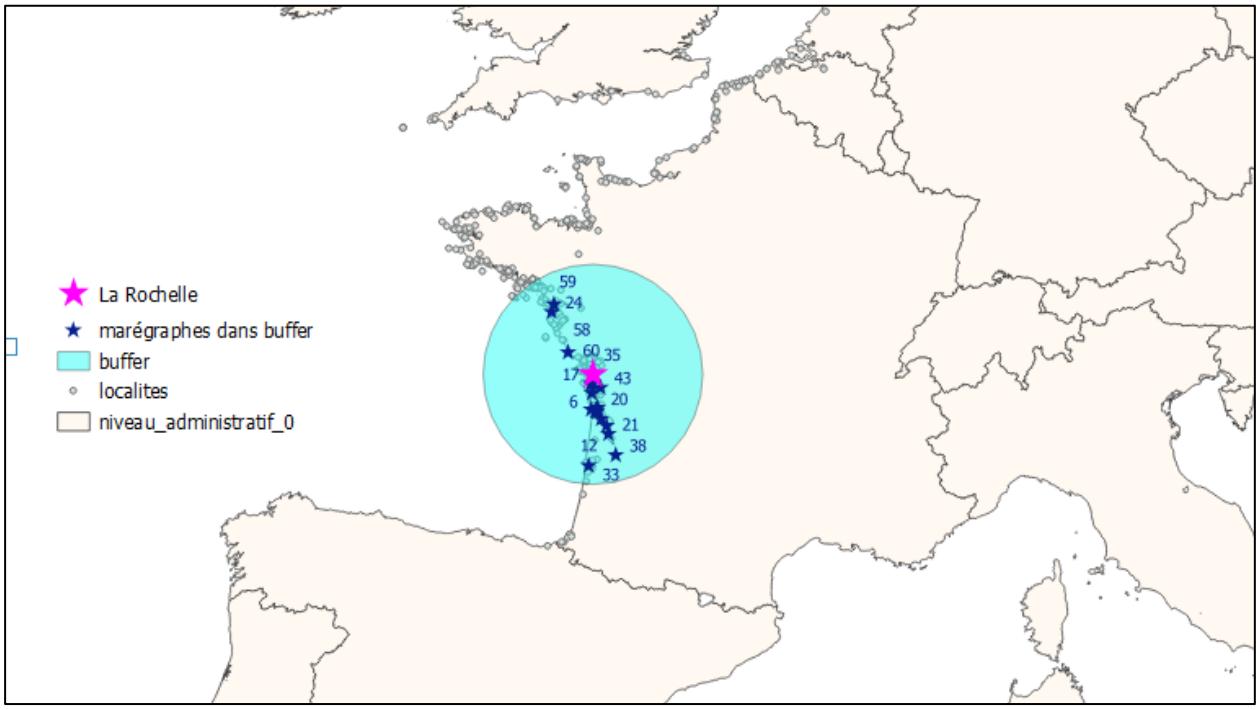
```
DROP TABLE IF EXISTS export.buffer;
CREATE TABLE export.buffer AS

SELECT
  row_number() over () AS gid,
  st_buffer(centre.loc_geom2, 200000) AS geom
FROM
  export.loc_193  AS centre
```

```
WHERE
  centre.loc_nom LIKE 'LA ROCHELLE';
```

Nous sélectionnons le point qui porte le nom `'LA ROCHELLE'`, nous appelons cette sélection `centre` et nous demandons de créer un rayon - buffer de 200 000 m autour de ce point.

Une fois cette table créée, nous pouvons procéder à la sélection des marégraphes (rappel : cette information est stockée dans le champ `loc_type` de la table `export.loc_l93`).
La fonction dont nous allons nous servir est `st_intersection()`

```
SELECT
  ville.loc_nom,
  st_intersection(ville.loc_geom2, buffer.geom)AS geom
FROM
  export.loc_l93 AS ville,
  export.buffer AS buffer
WHERE
  ville.loc_loctyp_id = 2;
```




Bien sûr, toutes ces opérations et traitements peuvent se faire directement dans un logiciel de SIG.

## 4.7 CONNEXION A LA BASE VIA R

Cette documentation présente les packages **RPostgreSQL** et **sqldf** qui permettent d'accéder au contenu de la base de donnée via **R**, ainsi que quelques exemples de traitement de données.

### 4.7.1 *CONNEXION A LA BASE DE DONNEES*

Les paramètres de connexion présentées ci-dessous permettent d'accéder à une base `tempete` stockée en local, sur le serveur `localhost` et le port `5434`.

```
library(RPostgreSQL)
library(sqldf)

# RPostgreSQL
db <- dbConnect(PostgreSQL(),
                host = "localhost",
                port = 5434,
                user = "tempete",
                password = "tempete",
                dbname = "tempete")


# sqldf
options(sqldf.RPostgreSQL.user ="tempete",
        sqldf.RPostgreSQL.password ="tempete",
        sqldf.RPostgreSQL.dbname = "tempete",
        sqldf.RPostgreSQL.host ="localhost",
        sqldf.RPostgreSQL.port =5434,
        sqldf.RPostgreSQL.encoding = "latin1")
```

### 4.7.2 *CHARGEMENT DE TABLES DANS LA BASE DE DONNEES*

#### 4.7.2.1 Via RPostgreSQL

Permet de charger le contenu d'une table via la fonction *dbReadTable*.

Chargement de la table *evenement*:

```
evenement <- dbReadTable(db, c("tempete", "evenement"))

head(evenement)

##   eve_id eve_nom eve_jour eve_mois eve_annee
## 1      1    <NA>       17       12      2004
## 2      2    <NA>       11        1      2007
## 3      3    <NA>       30        1      1877
## 4      4    <NA>       27        2      1903
## 5      5    <NA>       21        2      1993
## 6      7    <NA>       25        7      1867
```

db = le nom de la base de données

tempete = le nom du schéma

evenement = le nom de la table

### 4.7.2.2 Via sqldf

Ce package permet d'accéder aux données via le langage SQL. De plus, via ce même langage des manipulations sur la tables peuvent être faites (suppressions de tables, ajout de colonnes etc.).

Chargement de la table *evenement*

```
evenement2 <- sqldf("SELECT *
                     FROM tempete.evenement")

head(evenement2)
```

```
##   eve_id eve_nom eve_jour eve_mois eve_annee
## 1      1    <NA>       17       12      2004
## 2      2    <NA>       11        1      2007
## 3      3    <NA>       30        1      1877
## 4      4    <NA>       27        2      1903
## 5      5    <NA>       21        2      1993
## 6      7    <NA>       25        7      1867
```

L'exemple ci-dessus permet de charger toute la table. Si l'on souhaite charger qu'une partie de la table, il faut spécifier dans la requête SQL quelles colonnes on souhaite charger p.ex. *eve_id* et *eve_annee*:

```
evt_select <- sqldf("SELECT eve_id,
                            eve_annee
                     FROM tempete.evenement
                     ORDER BY eve_annee")

head(evt_select)
```

```
##   eve_id eve_annee
## 1    259      1507
## 2    558      1518
## 3    223      1525
## 4    510      1525
## 5    212      1530
## 6    460      1532
```

Il est également possible, de faire une sélection d'informations disponibles dans plusieurs tables, reliées entre-elles (i.e. tout type de requête SQL):

```
date_localite <-  sqldf("SELECT loc_nom,
                                evt_date
                         FROM tempete.localite,
                              tempete.evenement,
                              tempete.impact
```

```
                        WHERE impact.imp_loc_id = localite.loc_id AND
                              impact.imp_evt_id = evenement.evt_id")


head(date_localite)
```

```
##              loc_nom   evt_date
## 1        LA ROCHELLE 1518/08/10
## 2        LA ROCHELLE 1518/08/10
## 3 SAINT-MARTIN-DE-RE 1525/NA/NA
## 4 LA COUARDE-SUR-MER 1537/08/22
## 5     RIVEDOUX-PLAGE 1537/08/22
## 6          ARS-EN-RE 1591/02/24
```

## 4.7.3 *QUELQUES EXEMPLES*

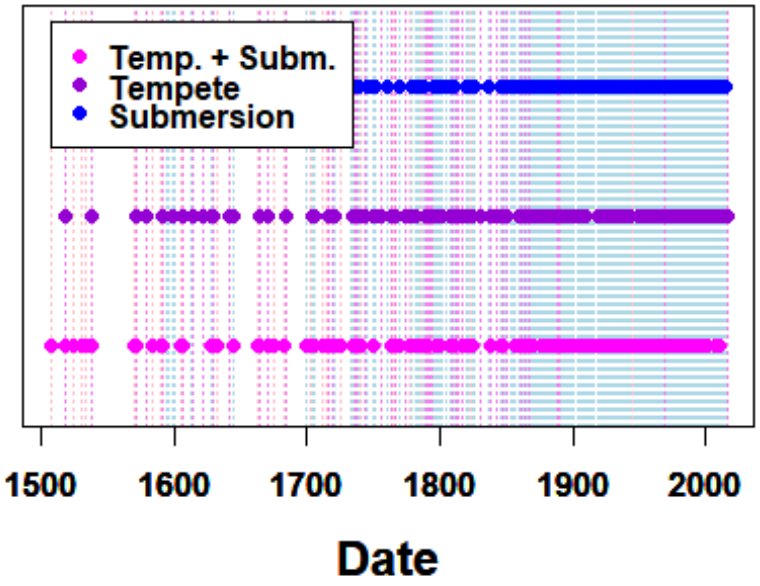
### 4.7.3.1 Exemples de requêtes

Ce script permet de réaliser une frise temporelle des événements disponibles dans la base de données.

```
## ========================================================================
# CHARGEMENT DE TABLES
## ========================================================================

eve_type <- sqldf("SELECT eve_id, eve_annee, eve_mois, eve_jour,
                          imp_imptyp_id
                   FROM tempete.evenement,
                           tempete.impact
                   WHERE impact.imp_eve_id = evenement.eve_id
                   ORDER BY eve_annee, eve_mois, eve_jour ASC")

imp_type <- sqldf("SELECT *
                  FROM tempete.impact_type ")

## ========================================================================
# TRAITEMENT DES DONNEES
## ========================================================================

type <- as.data.frame(matrix(0, nrow = length(eve_type$eve_id), ncol=6))
        colnames(type) <- c("eve_id", "TS", "S", "T", "Tsu", "final")

eve_unique <- unique(eve_type$eve_id)

type$eve_id <- eve_type$eve_id

for (i in 1:length(eve_unique)) {
  a <- as.numeric(eve_unique[i])
  index <- which(eve_type$eve_id == a)
  type$TS[i] <- length(which(eve_type$imp_imptyp_id[index] == "1"))
  type$S[i] <- length(which(eve_type$imp_imptyp_id[index] == "2"))
  type$T[i] <- length(which(eve_type$sou_evttype[index] == "3"))
  type$Tsu[i] <- length(which(eve_type$sou_evttype[index] == "4"))
}
```

```
for (i in 1:length(eve_unique)) {
  i <- as.numeric(i)
  imptyp <- colnames(type[which(type[i,] == max(type[i,2:5]))])
  type$final[i] <- imp_type$imptyp_id[which(imp_type$imptyp_libelle == imptyp
)]
}


## ========================================================================
# Format v3
## ========================================================================

#detecter champs vide et remplacer par NA
eve_type$eve_jour <- as.numeric(eve_type$eve_jour)
eve_type$eve_mois <- as.numeric(eve_type$eve_mois)

for (i in seq(1, length(eve_type$eve_id), 1)){
  if (is.na(as.character(eve_type$eve_jour[i]))){
    eve_type$eve_jour[i] <- "NA"
  }
}

for (i in seq(1, length(eve_type$eve_id), 1)){
  if (is.na(as.character(eve_type$eve_mois[i]))){
    eve_type$eve_mois[i] <- "NA"
  }
}

# jour
for (i in seq(1, length(eve_type$eve_id), 1)){
  if (nchar(eve_type$eve_jour[i]) ==  1) {
    eve_type$eve_jour[i] <- paste("0", eve_type$eve_jour[i], sep="" )
  } else {
    eve_type$eve_jour[i] <- eve_type$eve_jour[i]
  }
}

# mois
for (i in seq(1, length(eve_type$eve_id), 1)){
  if (nchar(eve_type$eve_mois[i]) == 1) {
    eve_type$eve_mois[i] <- paste("0", eve_type$eve_mois[i], sep="" )
  } else {
    eve_type$eve_mois[i] <- eve_type$eve_mois[i]
  }
}

eve_type$eve_date <- paste(eve_type$eve_annee, eve_type$eve_mois, eve_type$ev
e_jour, sep="/")
eve_type$eve_date <- gsub("NA", "01", eve_type$eve_date)
eve_type$eve_date2 <- as.POSIXct(eve_type$eve_date,
                                 format="%Y/%m/%d", tz="GMT")


index1 <- which(eve_type$imp_imptyp_id == 1)
index2 <- which(eve_type$imp_imptyp_id == 2)
index3 <- which(eve_type$imp_imptyp_id == 3)
```

```
## ========================================================================
# GRAPHIQUE
## ========================================================================

ini <- eve_type$eve_date2[1]
fin <- eve_type$eve_date2[length(eve_type$eve_date)]


#png("timeline.png", width = 1000, height = 600, units = "px")

plot(eve_type$eve_date2[index1], eve_type$imp_imptyp_id[index1], col="magenta
", pch=19,
     ylim=c(0.5,3.5), yaxt="n", font.lab = 2, font.axis=2,
     xlim=c(ini, fin),
     ylab = " ", xlab = "Date", cex.lab=1.5)
abline(v=eve_type$eve_date2[index1], col ="pink", lty = 3)
abline(v=eve_type$eve_date2[index2], col ="violet", lty = 3)
abline(v=eve_type$eve_date2[index3], col ="lightblue", lty = 3)


par(new=T)
plot(eve_type$eve_date2[index2], eve_type$imp_imptyp_id[index2], col="darkvio
let", pch=19,
     ylim=c(0.5,3.5), yaxt="n", #xaxt="n",
     xlim=c(ini, fin),
     ylab = " ", xlab = " ")
par(new=T)
plot(eve_type$eve_date2[index3], eve_type$imp_imptyp_id[index3], col="blue",
pch=19,
     ylim=c(0.5,3.5), yaxt="n", #xaxt="n",
     xlim=c(ini, fin),
     ylab = " ", xlab = " ")
legend(ini, 3.5,
       legend=c("Temp. + Subm.", "Tempete", "Submersion"),
       text.font = 2,
       pch = c(19,19),
       col = c("magenta", "darkviolet", "blue"),
       y.intersp=0.75)
par(new=T)
plot(eve_type$eve_date2[index1], eve_type$imp_imptyp_id[index1], col="magenta
", pch=19,
     ylim=c(0.5,3.5), yaxt="n", font.lab = 2, font.axis=2,
     xlim=c(ini, fin),
     ylab = " ", xlab = "Date", cex.lab=1.5)
```


Temp. + Subm.
Tempete
Submersion
1500
1600
1700
1800
1900
2000
Date

# 5 REFERENCES


[1] Garnier, E. and Surville, F. (2010). *La tempête Xynthia face à l'histoire*. Saintes, La Croix Vif.

[2] Audé, J.-L. (2006). *Chronique du climat en Poitou-Charentes Vendée: chronologie des phénomènes météorologiques et naturels du Moyen Age au XXème siècle*. Lonali.

[3] Lamb, H. and Frydendahl, K. (1991). *Historic Storms of the North Sea, British Isles and Northwest Europe*. Cambrigde, Cambridge University Press.

[4] Lang, M. and Coeur, D. (2014). *Les inondations remarquables en France*. Versailles, Quae.

[5] These Breilh, J.-F., 2014 - « Les surcotes et submersions marines dans la partie centrale du Golfe de Gascogne: les enseignements de la tempête Xynthia ».

[6] Letortu, P. (2013). *Le recul des falaises crayeuses haut-normandes et les inondations par la mer en Manche centrale et orientale : de la quantification de l'aléa à la caractérisation des risques induits.* Phd thesis, Université de Caen Basse-Normandie, 414 p.

[7] Perét, J. and Sauzeau, T. (2014). *Xynthia ou la mémoire réveillée. Des villages charentais et vendéens face à l'océan (XVIIIe-XXIe siècles)*. La Crèche, Editions Geste, La Crèche.

[8] Rapport BRGM, 2011 - « Contribution au recensement des submersions marines historiques liées aux tempêtes sur le littoal français métropolitain ».

[9] Rapport BRGM, 2013 - « Submersions marines historiques ».

[10] Rapport SHOM Daubord, C. and al., 2015 - « Rapport technique final du projet NIVEXT ».

[11] Roche, A. and al. (2014). Projet VIMERS: une typologie des tempêtes bretonnes pour prévoir l'impact des tempêtes à venir et mieux s'y préparer. In *Actes du colloque des XIIIe Journées Nationales Génie Côtier-Génie Civil, Dunkerque*, p. 2-4.

[12] Rapport Bodin, M. - Stage de fin d'étude à l'IRSN (M2 Université de La Rochelle), Août 2019 - « Valorisation des informations contenues dans Base de Données TEMPETES ET SUBMERSIONS HISTORIQUES ».

[13] Hamdi, Y. and al. (2018). Analysis of the risk associated with coastal flooding hazards: a new historical extreme storm surges dataset for Dunkirk, France. *Natural Hazards and Earth System Sciences*, **18**(12), p. 3383-3402.

[14] Athimon, E. and al. (2020). First methodological steps to define the reliability of historical documents and datasets used to study past storms, storm surges and surge levels. In *Workshop on Sea Level Data Archaeology*, 2020-03-10/12, Paris [Oral].

[15] Moreno, C. (2016). Rapport de stage : Analyse des dégâts de tempêtes historiques (Vimers) à la côte. *Rapport de Master 1, Université de La Rochelle*.

# ANNEXE 1 :
# Différences entre la v2 et la v3 de la base de données

## CONTEXTE

En début d'année 2020, trois étudiants de la licence professionnelle SIG proposée par l'Université de La Rochelle ont travaillé sur la base de données dans le cadre d'un projet de tutorat à réaliser pour valider leur formation. L'objectif du projet, encadré par l'IRSN, était double : dans un premier temps mettre à jour la structure de la base de données et dans un second temps développer pour l'IRSN une interface de consultation de base de données via un WebSIG. Le présent rapport documente brièvement les différences entre deux versions de la base de données (BD) TEMPETES ET SUBMERSIONS HISTORIQUES : la version v2, en date d'automne 2019 suite aux modifications apportées par Marceau Bodin dans le cadre de son stage de Master 2, et la version v3 développée par les étudiants de la licence professionnelle SIG.

La Figure 4 présente le modèle conceptuel de données (MCD) initial, transmis aux étudiants en Janvier 2020. Ce modèle contient 10 tables, dont 3 tables spatiales et une table intermédiaire. Les liens entre les tables présentent les relations. Le MCD développé par les étudiants et présenté sur la Figure 5.
De façon générale, la structure de la BD a été simplifiée, en supprimant un grand nombre de relations entre les tables, qui n'ont pas forcément raison d'être et en créant des tables « adjointes » pour alléger les tables et faciliter les requêtes. Visuellement, la nouvelle structure est plus claire (cf. Figure 4 et Figure 5).
De plus, tous les identifiants existants ont été remplacés par un identifiant numérique, de type serial, c'est-à-dire que l'identifiant est créé automatiquement lorsqu'un nouvel enregistrement est renseigné. Ainsi, un certain nombre d'informations redondantes est supprimé et la création d'un identifiant est automatisée, remplaçant les identifiants complexes mis en place auparavant.

Nous allons voir plus en détail ce qui a été modifié table par table, ainsi que les impacts que cela aura pour l'utilisateur. Certaines remarques peuvent sembler redondantes, mais sont nécessaires pour la compréhension des changements. Dans un premier temps, l'attention est portée sur les tables classiques, et dans un second temps sur les tables spatiales. Lorsque les descriptions portent sur des nouvelles tables, celles-ci sont présentées en jaune. Des propositions de modifications (→ indiquées en bleu clair sous l'item en question), ainsi que une liste de choses à faire (→ indiqués à la fin de chaque thématique / table en bleu marine) ont été identifiées lors d'un état des lieux réalisé en juin 2020 : les travaux menés entre cet état des lieux et la diffusion de ce présent rapport sont indiqués au cas par cas.

Dans ce bilan, les tables sont indiquées en minuscules avec la police `consolas` ayant la typologie suivante ; `evenement`. Les champs des tables, sont également indiqués à l'aide de la police `consolas` et marqué en *`italiques`*. Les nouvelles tables sont marquées par un astérisque *. Les tables spatiales sont marquées par un °.
N.B. : Les noms de tables ne portent pas d'accent.

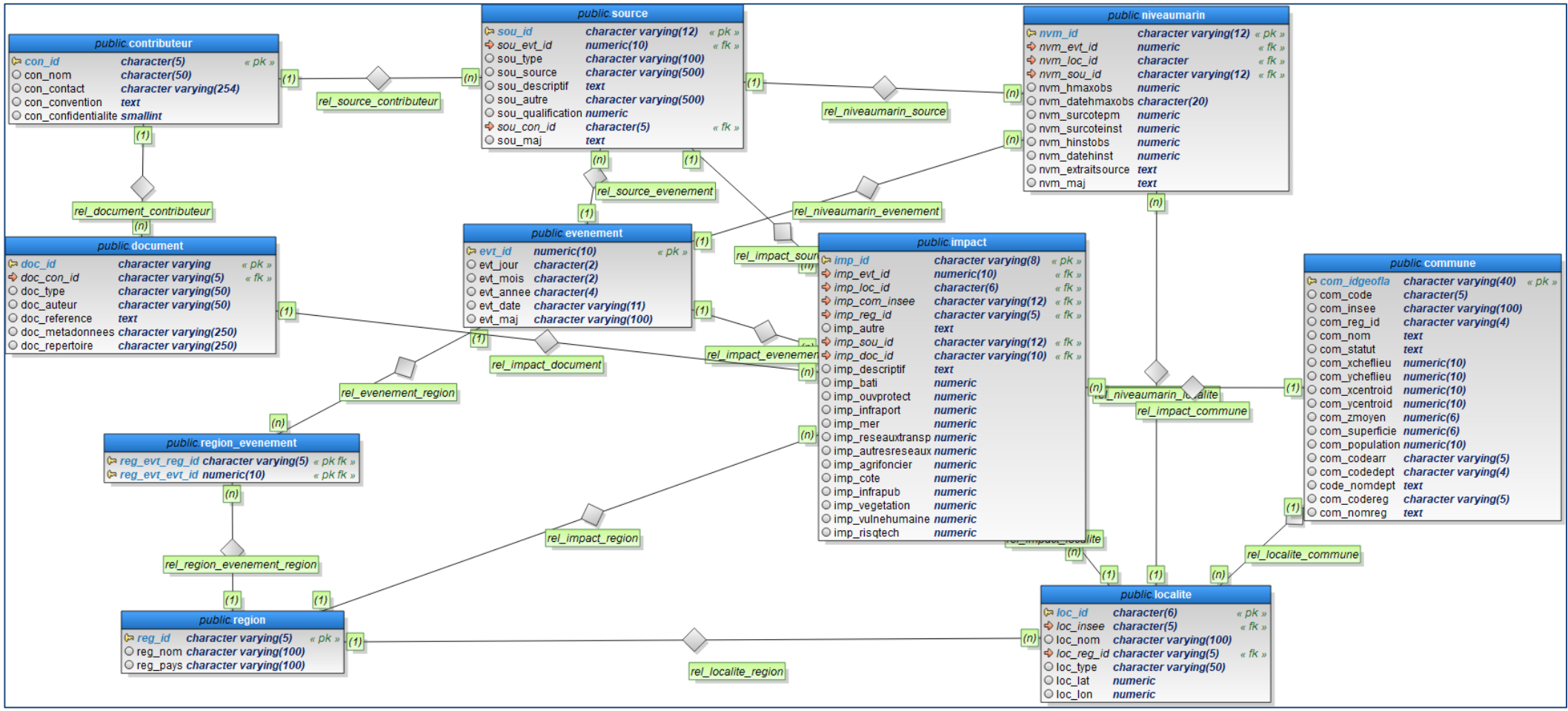



***Figure 4 : MCD de la Base de Données version 2.0 (Août 2019).***

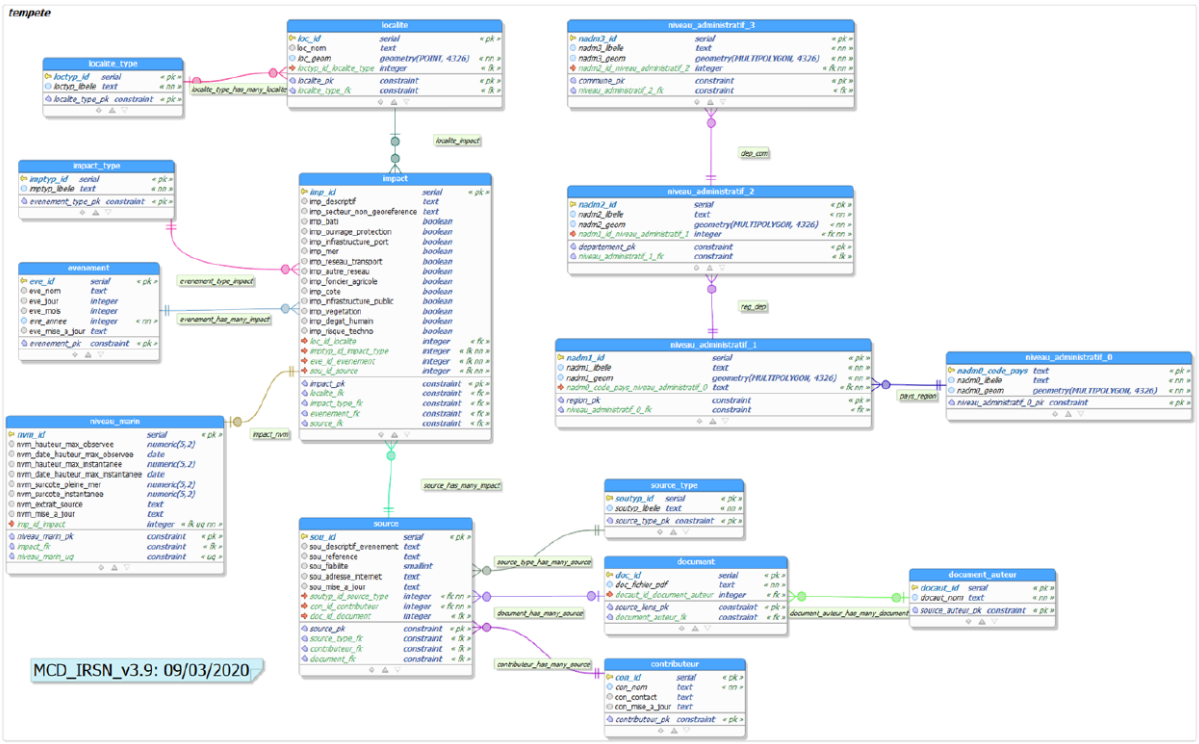



***Figure 5 : MCD de la Base de Données version 3.9 (Avril 2020).***

## ANALYSE

Table par table, nous allons analyser les différences entre ces deux versions. Les champs des tables des deux versions seront également comparés, permettant de voir les champs correspondants, les nouvelles apparitions ainsi que les champs supprimées. Le double astérisque, i.e. **, suivant le nom d'un champ, indique le renvoi vers de nouvelles tables. Dans les tableaux comparant les champs, les nouveaux champs (dans la v3) sont donnés en vert et les champs supprimés apparaissent en ~~rouge et barré~~ (v2).

### impact

La table impact devient table centrale de la base de données, et remplace ainsi l'ancienne table centrale, la table evenement, contenant la date d'occurrence d'un événement.

Dans cette nouvelle version de la base, la table impact devient élément central de la BD, car elle permet d'identifier les localisations impactées ainsi que les impacts recensés répondant ainsi au besoin exprimé, de mieux valoriser la donnée spatiale dans la base de données. Pour autant, la table ne contient pas d'informations géographiques, et ne peut donc pas être considérée comme table spatiale.

Les tables

- impact_type*
- evenement,
- source,
- niveau_marin et
- localite

sont directement reliés à la table impact. La table localite est la table spatiale, qui permet par la suite la localisation des informations, car elle possède un champ contenant l'information géographique (*Loc_geom*).

Tous les impacts, même s'ils ne sont pas localisables à l'aide d'une localité (i.e. une coordonnée X/Y dans un référentiel géographique) sont reportés dans la table impact, le champ *imp_id_localite* restant alors vide.

- Comme indiqué précédemment, seul le lien vers la localité exacte est gardé, cela correspond donc à la suppression des champs
  - reliant la table impact à la table commune (cette table est nommée niveau_administratifs_3 dans la nouvelle BD)
  - reliant la table impact à la table region (qui inclut les pays)

  De ce fait, un certain nombre d'informations géographiques est perdu : En supprimant les champs, il n'est désormais plus possible de géoréférencer les événements selon une commune ou une région / un pays (i.e. un polygone dans un référentiel géographique) si la localité exacte n'est pas disponible dans la source. Pour éclairer ceci, voici quelques chiffres :
  - La v2 contient 4057 lignes / impacts. 710 impacts ne sont pas géoréférencés à échelle d'une localité :
    - Dont 214 ont une indication de commune touchée
    - Dont 496 contiennent une information sur la région ou le pays touché
    - Et seulement 34 n'ont aucune information géographique
  - La v3 contient 4068 lignes / impacts[5]. 713 impacts ne sont pas géoréférencés à échelle d'une localité. Ces impacts ne peuvent alors pas être spatialisés (ni à échelle d'une commune, d'une région ou s'un pays).

| Impact | | |
|---|---|---|
| v3 | | v2 |
| *imp_id* | | *imp_id* |
| *imp_descriptif* | | *imp_descriptif* |
| *imp_secteur_non_georeference* | | *imp_autre* |
| *imp_bati* | | *imp_bati* |
| *imp_ouvprotect* | | *imp_ouvprotect* |
| *imp_infraport* | | *imp_infraport* |
| *imp_mer* | | *imp_mer* |
| *imp_reseauxtransp* | | *imp_reseauxtransp* |
| *imp_autresreseaux* | | *imp_autresreseaux* |
| *imp_agrifoncier* | | *imp_agrifoncier* |
| *imp_cote* | | *imp_cote* |
| *imp_infrapub* | | *imp_infrapub* |
| *imp_vegetation* | | *imp_vegetation* |
| *imp_vulnhumaine* | | *imp_vulnhumaine* |
| *imp_risqtechno* | | *imp_risqtechno* |
| *loc_id_localite* | | *imp_loc_id* |
| *imptyp_id_impact_type*** | | |
| *eve_id_evenement* | | *imp_evt_id* |
| *sou_id_source* | | *imp_sou_id* |
| | | ~~*imp_com_insee*~~ |
| | | ~~*imp_reg_id*~~ |

➔ Afin de ne pas perdre l'information sur une région et/ou un pays touché, un champ permettant de faire un lien vers la région / pays pourrait être ajouté à la base de données.

[5] N.B. L'écart entre le nombre d'enregistrement dans la table impact v2 et impact v3 a déjà été soulevé de la part des étudiants, mais l'origine n'est pas encore identifié.

/ ! \ Dans tous les cas, même en ajoutant ce champ, l'information de la source est plutôt « maigre » (elle ne mentionne pas une localité ou une commune précise), et cette information pourra uniquement guider l'utilisateur vers une région dans laquelle affiner les recherches.
Néanmoins, en faisant une requête sur un événement spécifique, l'information sur la région/le pays touché sortira comme résultat, – et n'est pas perdue.

- Le lien vers la table `localite` se fait via la clé primaire de cette table. Cependant, la simplification de la clé primaire de la table `localite`, ne permet plus de voir directement quelle commune était concernée par l'impact.
  Pour rappel, dans la v3, l'identifiant est un identifiant numérique, alors que dans la v2 l'identifiant était une combinaison de lettres et des chiffres comme indiqué ci-dessous :
  Les localités sont définies par un code composé :
  - d'un indicateur pays à trois caractères correspondant au code ISO Alpha 3
  - d'un code de ville composé de quatre ou cinq caractères et correspondant au code de la commune (France, Belgique) ou au postcode district (Grande Bretagne)
  - d'un code numérique à quatre chiffres. Si ce code commence par un 0, il s'agit d'un marégraphe, si le code commence par un 1, il s'agit d'une localité bien précise (centroïde de commune, port, église etc. mentionnée dans une source).

*Table 15: Exemples de code de localités.*

| | |
|---|---|
| **France** | |
| La Rochelle – La Pallice Marégraphe | **FR173000000** |
| La Rochelle | **FR173001000** |
| La Rochelle Promenade du Mail | **FR173001001** |
| **Belgique** | |
| Anvers | **BE110021000** |
| Zeebrugge Marégraphe | **BE310050000** |
| **Pays-Bas** | |
| Leyde | **NL05461000** |
| **Grande-Bretagne** | |
| Dover | **UKCT1600000** |
| Newlyn Marégraphe | **UKTR1800000** |

Ainsi, pour retrouver les communes impactées dans la v3, une requête spatiale doit être réalisée :
Exemple : Sélectionner les impacts dans la ville de Vannes.

```
SELECT *
FROM
  (
  SELECT loc_id AS vannes
  FROM tempete.localite, tempete.niveau_administratif_3
  WHERE st_intersects(loc_geom, nadm3_geom)
  AND nadm3_libelle LIKE 'Vannes'
  ) localite_vannes,
tempete.impact

WHERE  impact.imp_loc_id = vannes;
```

➔ Reprendre ligne par ligne les *imp_descriptif* - vérifier avec contenu de la table source et compléter (si possible) le champ *imp_impact_secteur_non_georef* : **FAIT**

➔ Identifier la raison pour laquelle le nombre d'enregistrements dans les tables impacts varie entre la v3 et la v2 : **Identification impossible. Mais dans le cadre de la du passage à la v3, mise à jour de la table impact**

➔ Reprendre ligne par ligne les sources et vérifier l'orthographe

`impact_type*`

Cette table a été créée, permettant d'identifier le type d'impact, en choisissant entre trois types prédéfinis :

- Tempête - concernant uniquement les impacts liés au vent
- Submersions - concernant les impacts de submersions
- Tempête + Submersions - concernant les impacts pour lesquels vents et inondations sont mentionnés.

| `impact_type` |
|---|
| `imptyp_id` |
| `imptype_libelle` |

➔ L'information perdue concerne les événements de tsunamis, mais très peu nombreux dans la BD v2, comptant seulement 3 événements.

➔ Renommer éventuellement la table, car `impact_type` peut facilement porter à confusion. En effet, en nommant la table `evenement_type` par exemple, il y aurait moins de doute sur le contenu de la table.

Dans la v2 cette information était référencée dans la table source et le champs *sou_evttype*.

La création de cette table, et l'ajout de l'information dans la table impact parait plus cohérent, car une source peut mentionner plusieurs impacts, mais c'est à échelle locale ou à l'échelle temporelle que le type peut varier.

Voici deux exemples de types tirés de Lemaire, A., 1857, Éphémérides dunkerquoises, revues, considérablement augmentées, Maillard et Vandenbussche, Dunkerque.[6]

Page 19 : 2. Février 1791 :

> *Marée formidable. Les quais sont inondés, les écluses dépassées, les digues Pollet et Lamorlière franchies en plusieurs points; des terrains considérables sont envahis par les eaux. L'Océan imitait les fureurs populaires de l'époque: les siennes furent heureusement de courte durée.*
>
> ➔ Submersion.

Page 62 : 2 Décembre 1807 :

> *Tempête horrible qui occasionne les plus épouvantables désastres en mer.*
>
> ➔ Tempête.

On constate que, dans une même source, plusieurs événements sont mentionnés, dont l'un fait clairement référence à un événement de submersion (2 Février 1791) et l'autre ne mentionne pas d'inondation sur les côtes (2 Décembre 1807).

[6] BNF, département Philosophie, histoire, sciences de l'homme, 8-LK7-2569, numérisé [en ligne sur Gallica].

evenement

Dans la v3 de la BD, la table evenement n'est plus que reliée à la table impact. C'est alors à travers de la table impact que le lien vers les autres tables est fait. En effet, bien que choisie comme table d'entrée d'origine, cette table ne contient finalement que la date des événements, le descriptif se trouvant alors dans la table source, et localisé ensuite à travers la table impact.

Comme pour les autres tables, l'identifiant est un numérique, facilitant le renseignement d'une nouvelle date. De plus, les champs *eve_mois* et *eve_jour* sont passé en type entier (integer), et peuvent rester vide, si le mois ou le jour ne sont pas précisés dans la source. Le champ contenant la date complète a également été supprimé, allégeant ainsi la table, et permettant de ne pas dupliquer l'information.
Dans le cas d'une requête, il sera tout à fait possible de créer un champ en concaténant les trois champs relatifs à la date pour reconstituer la date.
De plus, un champ permettant de renseigner le nom d'une tempête a été ajouté. Ainsi, des requêtes pourront être faites sur le nom de l'événement, qui, dans certains cas, est plus présent que la date exacte.

| evenement | | |
|---|---|---|
| v3<br>*eve_id*<br>*eve_nom:*<br>*eve_jour*<br>*eve_mois*<br>*eve_annee* | | v2<br>*evt_id*<br><br>*evt_jour*<br>*evt_mois*<br>*evt_annee*<br>~~*evt_date*~~ |

➔ Vérifier la présence de toutes les dates dans la table v2 et v3 : **FAIT**
➔ Ajouter, pour les tempêtes les plus récentes, le nom des tempêtes : **FAIT**

source

Seuls des changements mineurs ont été opérés dans le cadre de la mise à jour, concernant principalement le type de champs et la nomenclature des champs. Comme mentionné précédemment, le lien avec la table evenement est supprimé et se fait désormais via la table impact. Le type d'événement est également supprimé, car cette caractérisation se fait dans la table impact.

| source | | |
|---|---|---|
| v3<br>*sou_id*<br>*sou_descriptif_evenement*<br>*sou_reference*<br>*sou_fiabilite*<br>*sou_adresse_internet*<br>*sou_mise_a_jour*<br>*soutyp_id_source_type***<br>*con_id_contributeur*<br>*doc_id_document* | | v2<br>*sou_id*<br>*sou_descriptif*<br>*sou_source*<br>*sou_qualification*<br>*sou_autre*<br><br>*sou_qualification*<br>*sou_con_id*<br>*sou_doc_id*<br>~~*sou_evt_id*~~<br>~~*sou_evttype*~~ |

➔ Mettre à jour le type de source selon les propositions retenues dans le GT: **FAIT**.

### source_type*

Cette nouvelle table permet la distinction via un identifiant numérique du type de la source, simplifiant également les requêtes. Actuellement le type de source référence des informations variées et pas très cohérentes, repris tel quel de la v2 de la base de données, cf. Figure 6.

| soutyp_id [PK] integer | soutyp_libelle text |
|---|---|
| 1 | Document d'archive |
| 2 | Proceeding |
| 3 | Information sur les risques majeurs |
| 4 | Rapport |
| 5 | PPRN rochefort |
| 6 | Bulletin d'assocation |
| 7 | Receuil historique |
| 8 | Mémoire M1 |
| 9 | Atlas des risques littoraux |
| 10 | Blog |
| 11 | Annales |

***Figure 6 : Extrait des types de sources actuellement présents dans la v3 de la BD.***

Des discussions au sein du GT ont permis de définir plus clairement les types de source, et ont abouti à la création de 5 types de documents :

- SP : Source primaire,
- SSM Source seconde main,
- LS Littérature scientifique,
- LT Littérature technique et
- RI Référence Intraçable.

| Source_type |
|---|
| *soutyp_id* |
| *soutype_libelle* |

➔ Pour être cohérent avec les définitions des types de source, une modification de la table intégrant uniquement les cinq types de sources retenus dans le cadre du GT, devra être envisagée : **FAIT**.

### contributeur

La table contributeur est quasi identique à celle déjà présente dans la v2. En effet, la BD publique ne contient que des informations publiques, et donc le champ *con_confidentialite* semble superflu.

| Contributeur | |
|---|---|
| v3 | v2 |
| con_id | con_id |
| con_nom | con_nom |
| Con_contact | con_contact |
| Con_mise_a_jour | con_convention |
| | ~~con_confidentialite~~ |

➔ Dans le cadre de la BD IRSN ou de la BD GT, le rétablissement de ce champ pourrait être envisagé.

## document

La table document était un ajout lors du passage de la v1 à la v2 de la base de données, à l'occasion du stage de M. Bodin à l'IRSN au cours de l'été 2019. A cette époque, des champs faisant doublons avec les champs présents dans la table source avaient déjà été identifiés. Le projet tutoré était alors l'occasion de supprimer ces doublons pour ne garder que les champs utiles. Ainsi, quatre champs ont été supprimés lors du passage à la v3.

| Document | | |
|---|---|---|
| v3 | | v2 |
| doc_id | | doc_id |
| doc_fichier_pdf | | doc_repertoire |
| doc_aut_id_document_auteur** | | |
| | | ~~doc_type~~ |
| | | ~~doc_con_id~~ |
| | | ~~doc_type~~ |
| | | ~~doc_auteur~~ |

## document_auteur*

La table document_auteur a été ajoutée à la base de données, elle contient uniquement deux champs :

| document_auteur |
|---|
| *docaut_id* |
| *docaut_nom* |

➔ Ces informations sont déjà présentes dans la table source et le champ *sou_reference*. La question de la nécessité de cette table peut alors se poser.

## niveau_marin

La table niveau_marin ne présente que quelques changements en lien avec la simplification de la base de données : trois champs reliant anciennement des tables, ont été supprimés. Ce lien est cependant toujours présent, mais se fait à travers la table impact, comme vu précédemment.

| Niveau_marin | | |
|---|---|---|
| v3 | | v2 |
| *nvm_id* | | *nvm_id* |
| *nvm_hauteur_max_observee* | | *nvm_hmaxobs* |
| *nvm_date_hauteur_max_observee* | | *nvm_datehmaxobs* |
| *nvm_date_max_instantanee* | | *nvm_surcotepm* |
| *nvm_hauteur_max_instantanee* | | *nvm_surcoteinst* |
| *nvm_surcote_pleine_mer* | | *nvm_datemaxinst* |
| *nvm_surcote_instantanee* | | *nvm_hmaxinst* |
| *nvm_extraitsource* | | *nvm_extraitsource* |
| *nmv_mise_a_jour* | | ~~*nvm_evt_id*~~ |
| | | ~~*nvm_loc_id*~~ |
| | | ~~*nvm_sou_id*~~ |

## localite°

La table localité a subi peu de changement. Comme pour les autres tables, l'identifiant a été modifié. Le type de localité est également passé en numérique, une table associée a été ajoutée pour les définir. Les champs contenant les coordonnées X et Y des localités ont été supprimé, l'information était déjà stockée dans la géométrie de la table (*Loc_geom*).

| Localité | |
|---|---|
| v3 | v2 |
| *loc_id* | *loc_id* |
| *loc_nom* | *loc_nom* |
| *loc_geom* | *loc_geom* |
| *loctyp_id_locality_type*** | *loc_type* |
| | ~~*loc_lat*~~ |
| | ~~*loc_lon*~~ |
| | ~~*loc_reg_code*~~ |
| | ~~*loc_insee*~~ |

➔ Une réflexion devra être menée pour harmoniser la nomenclature des localités. Les localités précises dans une commune, devront-elles porter le nom de la ville en plus ? Ainsi le Quai de la Cunette à Dunkerque pourrait être nommé « DUNKERQUE – QUAI DE LA CUNETTE ». Ceci permettrait de distinguer plusieurs localités pouvant porter le même nom. A Saint-Nazaire, en Janvier 1877, la « rue neuve » fut submergée par la mer et l'on peut facilement imaginer qu'une rue neuve existe dans la plupart des communes françaises: **FAIT**.

localite_type*

La table localité a été créée permettant l'identification du type de localité impactée.
Actuellement, les types sont présents dans la table :

Marégraphe = point du marégraphe
Ville = centroïde de polygone
Localité = localité ou lieu-dit

| localite_type |
|---|
| *loctyp_id* |
| *loctyp_libelle* |

➔ Vérifier que les types de localités soient bien attribués : **FAIT**

➔ Mener une réflexion sur d'autres types à intégrer comme par exemple un marais ou une digue

Les quatre tables suivantes sont des tables spatiales de type polygone. Chaque table contient un le contour d'une unité administrative allant du pays à la commune. Afin d'avoir des données homogènes, les données proviennent du site web Eurostats7, qui met gratuitement à disposition les couches géographiques (shapefiles). Des liens entre les différentes tables ont été ajoutés. Cependant, comme il n'existe plus de lien entre les couches de type polygone et les localités (géométrie de type point) le lien est désormais uniquement spatial.

[7] https://ec.europa.eu/eurostat/fr/web/gisco/geodata/reference-data/administrative-units-statistical-units/nuts

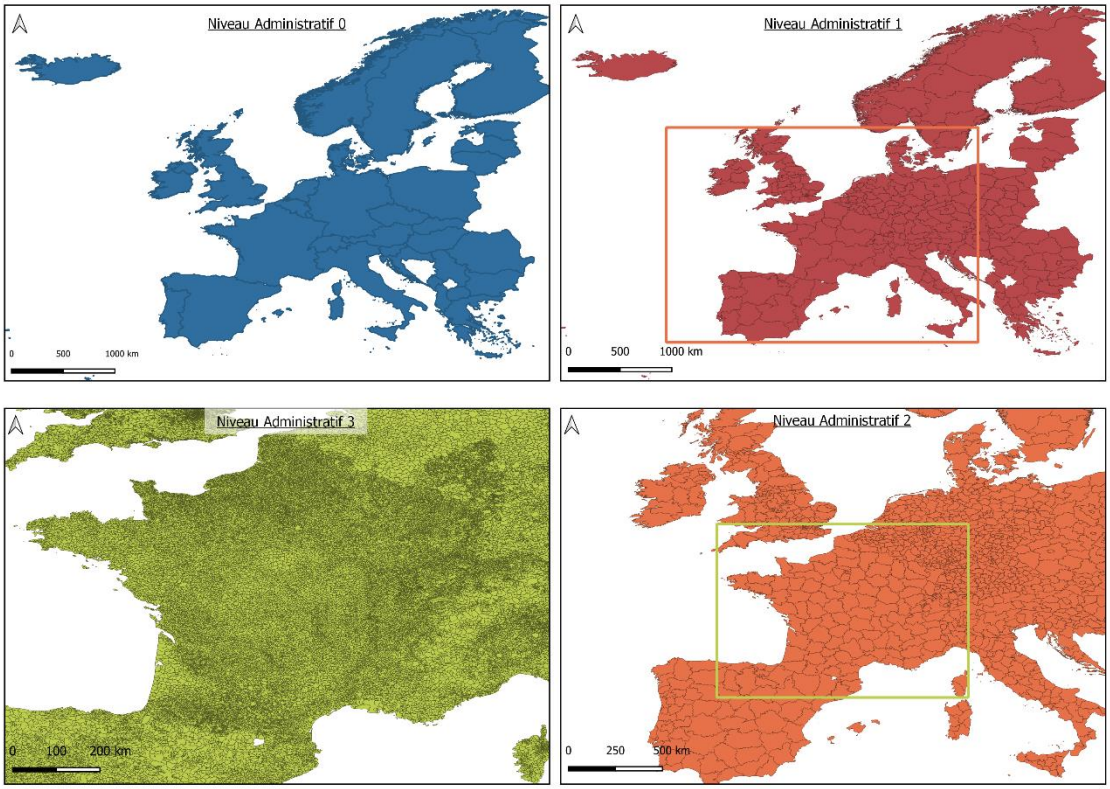



***Figure 7 : Les nouveaux découpages administratifs disponibles dans la v3.***

La Figure 7 présente les nouveaux découpages administratifs disponibles dans la v3. Le découpage correspond au découpage NUTS – Nomenclature des Unités Territoriales Statistiques (NUTS) – disponible sur le site Eurostat 8.

### niveau_administratif_0°

Le niveau administratif 0 correspond aux limites des pays européens.

| niveau_administratif_0 |
|---|
| *nadm0_code_pays* |
| *nadm0_libelle* |
| *nadm0_geom* |

### niveau_administratif_1°

Le niveau administratif de type 1 corresponds aux régions des pays. Ainsi pour la France on retrouve les régions administratives, les régions présentes dans la version précédente disparaissent. Concrètement cela représente seulement la perte de la région « Poitou-Charentes», qui depuis 2016, a été fusionnée avec la région « Aquitaine » en donnant la nouvelle grande région « Nouvelle Aquitaine ». Cela est encore une fois en phase avec la simplification de la base de données, et l'idée de rester sur des découpages « classiques ».

| niveau_administratif_1 |
|---|
| *nadm1_id* |
| *nadm1_libelle* |
| *nadm1_geom* |
| *nadm1_code_pays_niveau_administratif_0* |

[8] Eurostat, NUTS, https://ec.europa.eu/eurostat/fr/web/gisco/geodata/reference-data/administrative-units-statistical-units/nuts (consulté le 09/06/2020).

### niveau_administratif_2°

Le niveau administratif 2, correspond aux départements pour la France.

| niveau_administratif_2 |
| --- |
| *nadm2_id* |
| *nadm2_libelle* |
| *nadm2_geom* |
| *nadm1_id_niveau_administratif_1* |

### niveau_administratif_3°

Enfin, le niveau le plus fin correspond aux communes.

| niveau_administratif_3 |
| --- |
| *nadm3_id* |
| *nadm3_libelle* |
| *nadm3_geom* |
| *nadm1_id_niveau_administratif_2* |

- ➔ Le lien avec le COG (Code Officiel Géographique) des communs de l'INSEE est perdu. Réfléchir à un ajout ?
- ➔ / ! \ Le COG existe uniquement pour les communes françaises, et n'est pas disponible pour les communes des autres pays européens.

## CONCLUSION ET PERSPECTIVES

La structure de la base de données v3 élaborée dans le cadre du projet tutoré dataStorm réalisée par les étudiants de la LUP SIG de La Rochelle est une amélioration et simplification de la base de données v2.
Le côté spatial de la base de données et clairement mis en avant. Cela signifie également que pour faire les requêtes, des requêtes spatiales via postGIS doivent être faites, nécessitant des connaissances plus poussées du langage SQL.

Au cours de la comparaison, des points ont été soulevés ; les points principaux ainsi que des compléments sont repris dans la liste suivante :

Table `impact`

- ➔ Reprendre ligne par ligne les imp_descriptif – vérifier avec contenu de la table SOURCE et compléter (si possible) le champ imp_impact_secteur_non_georef : **FAIT**
- ➔ Identifier la raison pour laquelle le nombre d'enregistrement dans les tables impacts varie entre la v3 et la v2 : **Identification impossible. Mais dans le cadre de la du passage à la v3, mise à jour de la table impact.**
- ➔ Reprendre ligne par ligne les sources vérifier l'orthographe
  - ➔ Afin de ne pas perdre l'information sur une région et/ou un pays touché, un champ permettant de faire un lien vers la région / pays pourrait être ajoutée à la base de données.
    / ! \ Dans tous les cas, même en ajoutant ce champ, l'information de la source est plutôt « maigre » (car elle ne mentionne pas une localité ou une commune précise), et cette information pourra uniquement guider l'utilisateur vers une région dans laquelle affiner les recherches.
    Néanmoins, en faisant une requête sur un événement spécifique, l'information sur la région/le pays touché sortira comme résultat, – et n'est pas perdue.

Table `impact_type*`

- ➔ L'information perdue concerne les événements de tsunamis, mais très peu nombreux dans la BD v2, comptant seulement 3 événements.
- ➔ Renommer éventuellement la table, car `impact_type` peut facilement porter à confusion. En effet, en nommant la table `evenement_type` par exemple, il y aurait moins de doute sur le contenu de la table.

Table `evenement`

- ➔ Vérifier la présence de toutes les dates dans la table v2 et v3 : **FAIT**
- ➔ Ajouter, pour les tempêtes les plus récentes, le nom des tempêtes : **FAIT**

Table `source`

- ➔ Mettre à jour le type de source selon les propositions retenues dans le GT : **FAIT**.

Table `source_type*`

- ➔ Pour être cohérent avec les définitions des types de source, une modification de la table intégrant uniquement les cinq types de sources retenus dans le cadre du GT, devra être envisagée: **FAIT**.

Table `localite`

- ➔ Une réflexion devra être menée pour harmoniser la nomenclature des localités. Les localités précises dans une commune, devront-elles porter le nom de la ville en plus ? Ainsi le Quai de la Cunette à Dunkerque pourrait être nommé « DUNKERQUE – QUAI DE LA CUNETTE ». Ceci permettrait de distinguer plusieurs localités pouvant

porter le même nom. A Saint-Nazaire, en 1877, la « rue neuve » fut submergée par la mer et l'on peut facilement imaginer qu'une rue neuve existe dans la plupart des communes françaises : **FAIT**.

Table `localite_type*`

➔ Vérifier que les types de localités soient bien attribués : **FAIT**

➔ Mener une réflexion sur d'autres types à intégrer comme par exemple un marais ou une digue

Table `niveau_administratif_3`

➔ Le lien avec le COG (Code Officiel Géographique) de l'INSEE est perdu. Réfléchir à un ajout ?

Unités Littorales

Dans la mise à jour de la base de données, le découpage du littoral français en unités littorales réalisé par le Conservatoire du littoral, n'a pas été inclus. Cependant d'après le rapport IRSN/2020-00339 ce découpage est le plus pertinent si l'on souhaite faire une analyse de la vulnérabilité du littoral au risque de tempêtes et submersions.

➔ Ainsi, l'intégration de ce découpage pourrait être envisagée dans une future version de la base de données ; des requêtes spatiales pourront alors être mises en place assez facilement.

En plus de la mise à jour de la structure de la base de données, le développement d'un WebSIG à travers une interface Leaflet a été réalisé par les étudiants. Cette interface, très intuitive et facile à prendre en main, permet de s'affranchir du langage SQL et de l'écriture de requêtes spatiales, car l'interrogation de la base se fait via une interface cartographique contentant également des formulaires de requêtes. A ce jour, cette interface est uniquement disponible à l'IRSN.

# ANNEXE 2 :
# Solution des requêtes

1. Compter les submersions qui ont touché Brest.

```
SELECT
  count(nvm_id)
FROM
  tempete.impact,
  tempete.localite,
  tempete.niveau_marin
WHERE
  impact.imp_id = niveau_marin.nvm_id  AND
  localite.loc_id = impact.imp_loc_id AND
  localite.loc_nom like 'BREST%'
```

2. Quelles sont les villes touchées par une tempête - ex. 9 Janvier 1924, avec les infos complémentaires

```
SELECT
  loc_nom,
  imp_descriptif
FROM
  tempete.impact,
  tempete.localite,
  tempete.evenement,
  tempete.niveau_marin
WHERE
  evenement.eve_id = impact.imp_eve_id  AND
  localite.loc_id = impact.imp_loc_id AND
  evenement.eve_annee = 1924 AND
  evenement.eve_mois = 1 AND
  evenement.eve_jour = 9
```

3. Quelles sont les dates de tempêtes ayant causées une submersion supérieure à 50 cm à La Rochelle

```
SELECT
  loc_nom,
  eve_annee,
  eve_mois,
  eve_jour,
  nvm_date_hauteur_max_observee,
  nvm_hauteur_max_observee,
  nvm_surcote_pleine_mer,
  nvm_date_hauteur_max_instantanee,
  nvm_hauteur_max_instantanee,
  nvm_surcote_instantanee
FROM
  tempete.impact,
  tempete.localite,
```

```
  tempete.evenement,
  tempete.niveau_marin
WHERE
  evenement.eve_id = impact.imp_eve_id  AND
  localite.loc_id = impact.imp_loc_id AND
  impact.imp_id = niveau_marin.nvm_imp_id AND
  localite.loc_nom LIKE '%LA ROCHELLE%' AND
  niveau_marin.nvm_surcote_pleine_mer >= 50
```

4. Trouver les sources pour un événement

```
SELECT
  sou_reference,
  sou_conservationcote,
  sou_adresse_internet
FROM
  tempete.source,
  tempete.evenement,
  tempete.impact
WHERE
  evenement.eve_id = impact.imp_eve_id AND
  source.sou_id = impact.imp_sou_id AND
  evenement.eve_annee = 1924 AND
  evenement.eve_mois = 1 AND
  evenement.eve_jour = 9
GROUP BY
  (sou_reference, sou_conservationcote, sou_adresse_internet)
```

5. Compter les sources pour un événement

```
SELECT
  count((sou_reference, sou_conservationcote, sou_adresse_internet))
FROM
  tempete.source,
  tempete.evenement,
  tempete.impact
WHERE
  evenement.eve_id = impact.imp_eve_id AND
  source.sou_id = impact.imp_sou_id AND
  evenement.eve_annee = 1924 AND
  evenement.eve_mois = 1 AND
  evenement.eve_jour = 9
```