Faculté des sciences
Département d'informatique

Document de passation M2– J2

Réalisé par:

GAGJ2750 - Jonathan Gagnon

BEAR2809 - Raphaël-Jonathan Beaupré

DIAB0981 - Boubacar Siddikh Diallo

ELLY7998 - Yassine Ellatifi

Remis à :
Luc Lavoie

Dans le cadre de l'activité pédagogique
IGE487 – Modélisation de base de données

Date de remise : 14 novembre 2025

TABLE DES MATIERES

Introduction

Le présent document a pour objectif de présenter les modèles logique et conceptuel de notre base de données, en vue de les partager avec d'autres groupes dans le cadre de la création d'un entrepôt de données. Dans un premier temps, ce document expose de manière claire la structure conceptuelle et logique de la base. Par la suite, il met en avant les fonctions IMM au moyen d'un glossaire indexé. Étant donné que les lecteurs ont déjà travaillé sur ce sujet, nous avons jugé pertinent d'expliquer les principales modifications ainsi que les choix d'implémentation, afin de faciliter la compréhension pour les futurs lecteurs.

Évolution du document

Version	Date	Auteur(s)	Modifications
1.0	2025-11-10	-	Première version complète

Notation

DFNC : dépendance fonctionnelle NON induite par les clés candidates.

FNBC : forme normale de Boyce-Codd.

5FN : cinquième forme normale.

FNPJ : forme normale de projection-jointure

Cas d'étude

Voici les tables qui étaient fournies au début du projet:

Schéma relationnel du schéma herbivorie original

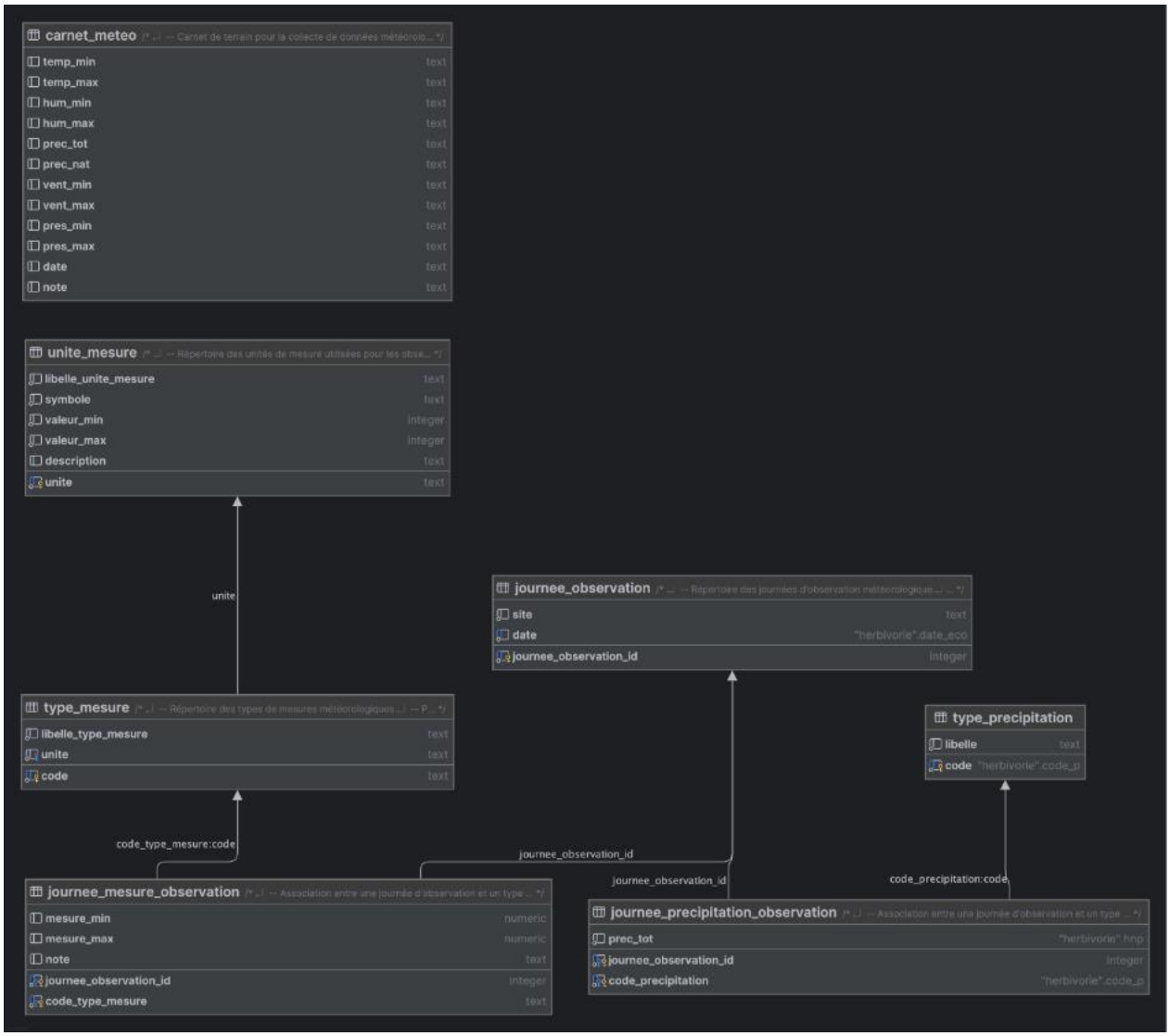


Schéma relationnel du schéma météo original

Modélisation conceptuelle (modèle entité-relation)

Un modèle entité-relation du projet herbivorie a été élaboré à partir de l'analyse du cas d'étude et de la compréhension des besoins.

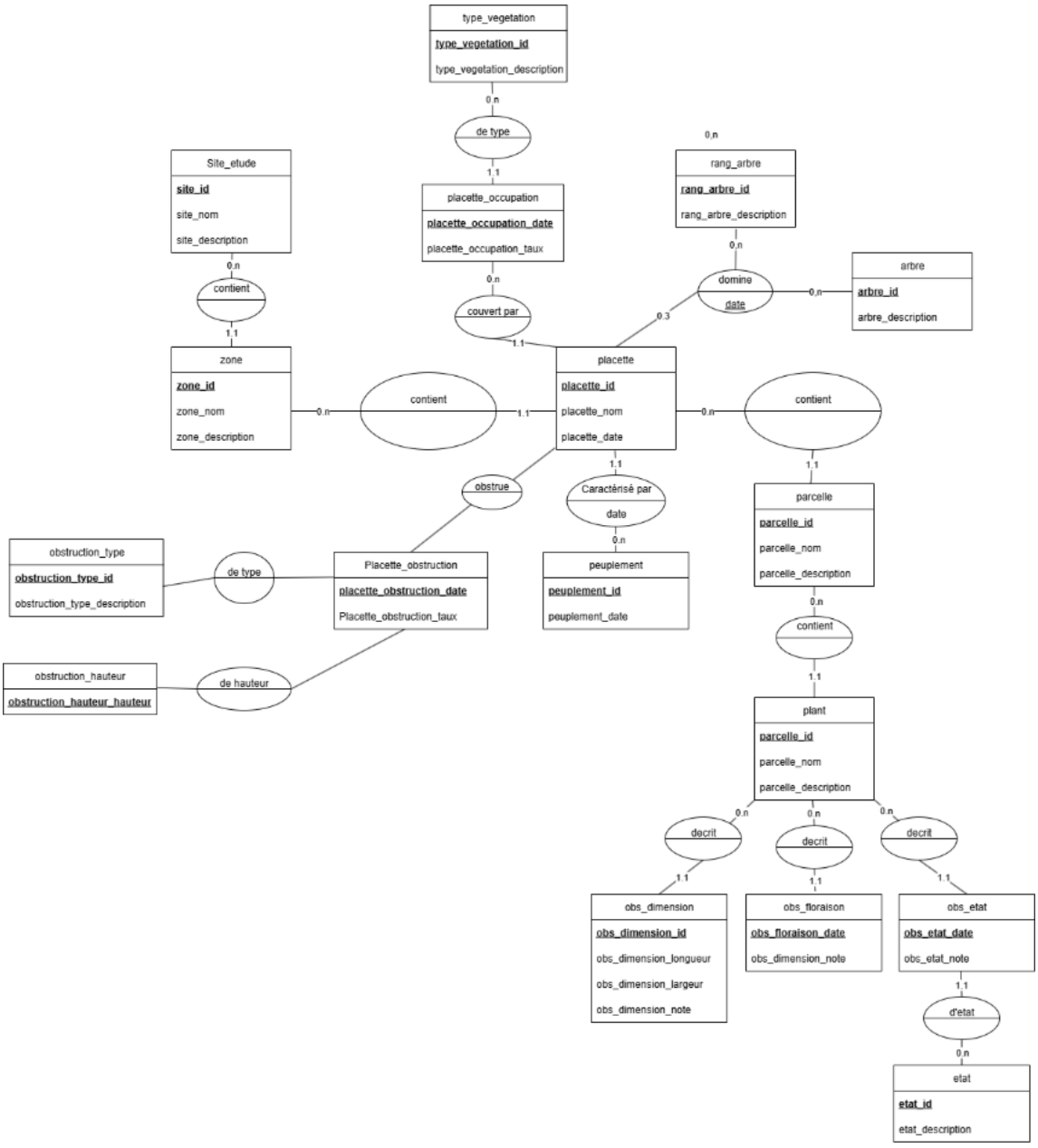


Schéma entité-relation du schéma herbivorie

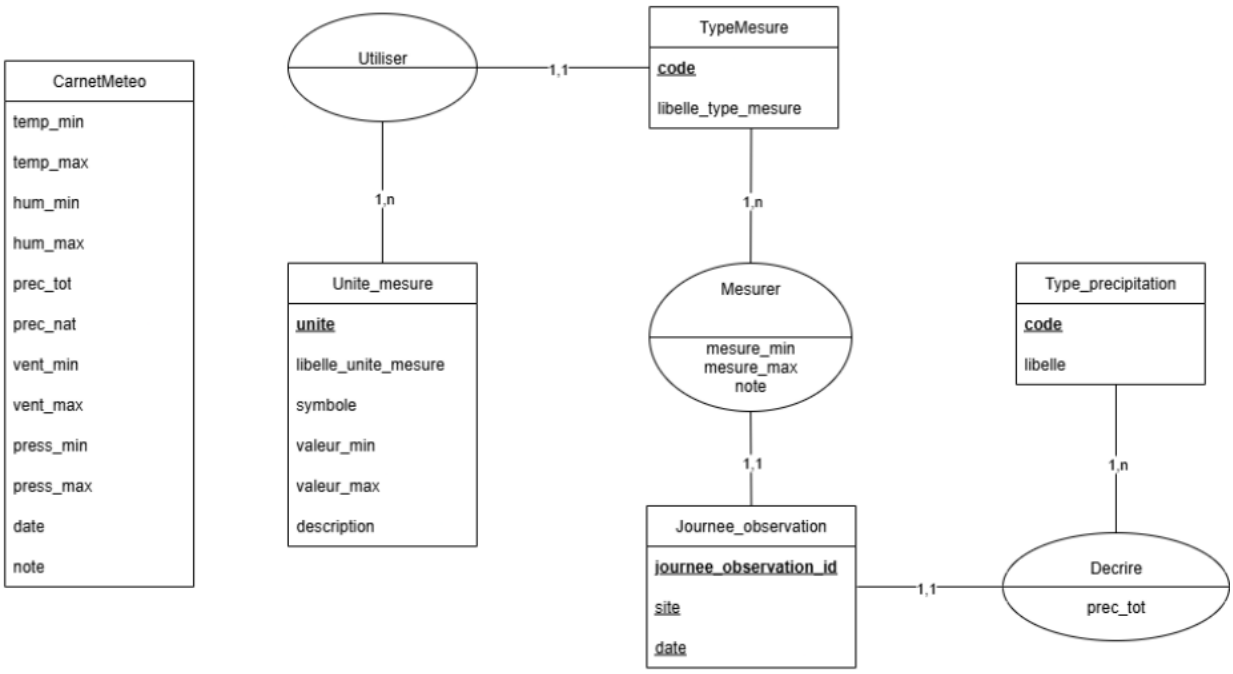


Schéma entité-relation du schéma météo

Modification et justification

Taux

Une table taux contenait des catégories et un intervalle pour caractériser un taux en pourcentage. Cette table pouvait causer plusieurs problèmes. Premièrement, si une modification était appliquée au min ou max des catégories, toutes les valeurs prisent précédemment pouvaient devenir erronée. De plus, enregistrer les valeurs précises directement dans les tables fait beaucoup plus de sens d'un point de vue statistique.

Site d'étude

Une table site d'étude identifié par un id et caractérisée par un nom et une description a été ajoutée pour permettre d'identifier à quel site une placette appartient.

Zone d'étude

Une table zone d'étude identifié par un id et caractérisée par un nom et une description a été ajoutée pour permettre d'identifier à quel site une placette appartient. Une zone se trouve dans un site.

Placette

La table placette contenait beaucoup d'information qui pouvait être stocké d'une façon bien meilleure. Un champ zone_id a été ajouté dans la placette puisqu'elle se trouve dans une zone.

Premièrement, tous les attributs d'obstruction peuvent être stockés dans une même table en étant caractérisés par une date d'observation, un taux d'obstruction (%) et un type d'obstruction (feuillu, conifère) et une hauteur (1m, 2m). Le total peut facilement être calculé à partir des valeurs de conifère et de feuillu a une hauteur donnée pour une date donnée.

Deuxièmement, le même problème se produisait pour les taux d'occupation au sol pour les trois types. Une table a donc été ajoutée et est caractérisé par un type de plante (mousse, fougère, graminée), un taux d'occupation (%) et une date d'observation.

Finalement, plutôt que de stocker les trois espèces d'arbres dominants directement dans la table placette, une table a été ajoutée et est caractérisée par une espèce d'arbre, un rang et une date d'observation.

Parcelle

Une table parcelle identifié par un id et caractérisée par un nom et une description a été ajoutée pour permettre d'identifier à quel site une placette appartient. Une zone se trouve dans un site.

Type d'obstruction

Une table type d'obstruction identifiée par un identifiant et caractérisé par un nom et une description a été ajoutée.

Hauteur d'obstruction

Une table hauteur d'obstruction identifié par une hauteur a été ajoutée pour identifier les hauteurs d'obstruction qu'il était possible de caractériser dans une placette.

Observation de la floraison

Le champs booléen "fleur" pour caractériser si un plant possédait une fleur a un instant donné a été retiré. On ajoute maintenant une donnée s'il y a présence de fleur et on ne fait rien si un plant n'a pas de fleur. Il est toujours possible de connaitre si un plant a une fleur a une date précise ou non sans le champ.

Les indices

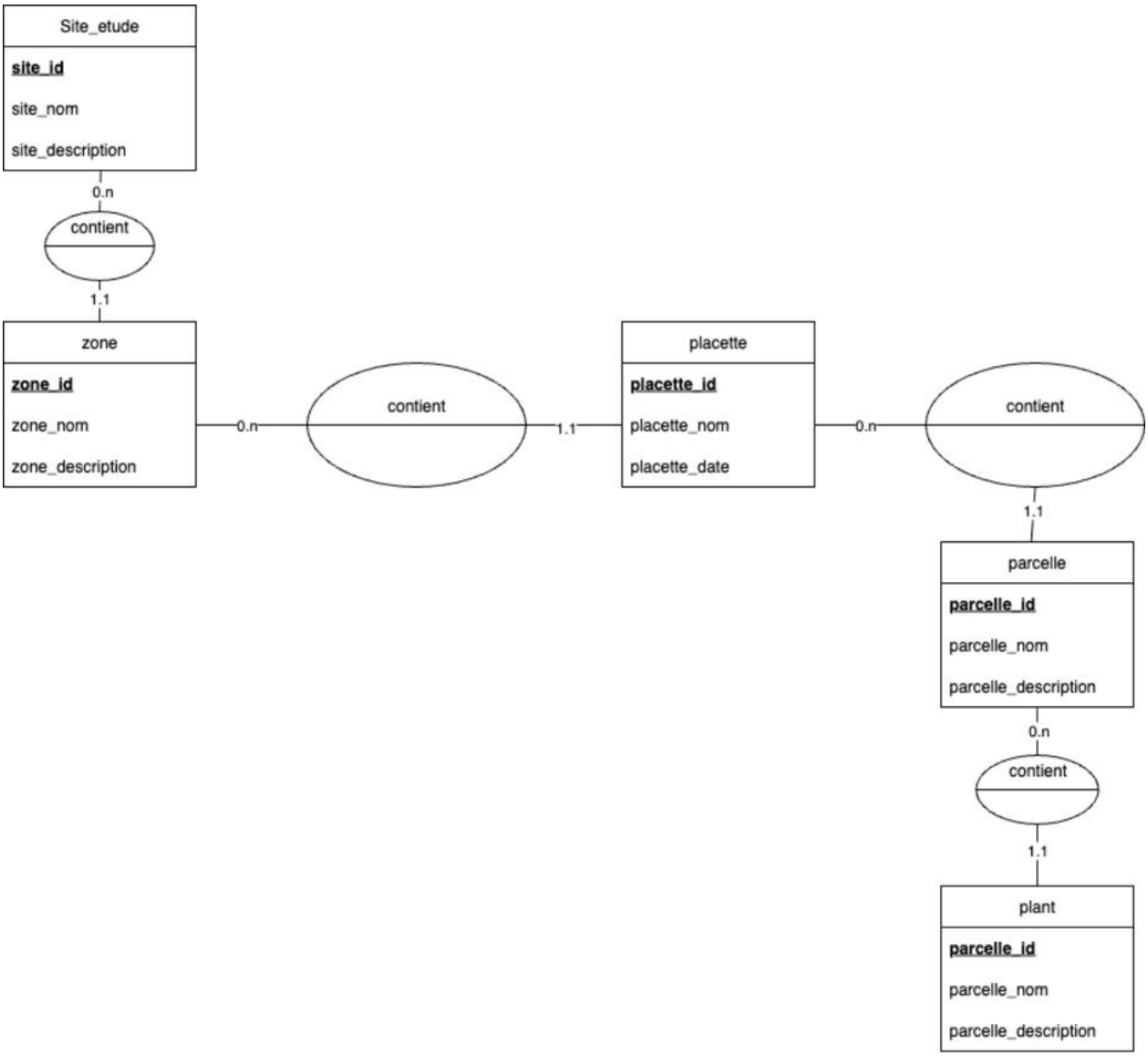
Initialement les indices de la table plant était organisé sous la forme suivante :

‘MMA9168’

Il était possible d’effectuer une subdivision de la clé pour obtenir les informations d’appartenance de chacun des éléments. Concrètement, à partir de l’indice, on pouvait savoir à quel site, zone et placette un plant appartenait. De plus, il était possible de déterminer à quelle parcelle appartenait un plant à partir de l’attribut parcelle.

Nous avons considéré ce traitement comme incorrect, avec cette solution il était, par exemple, très compliqué de retrouver l’appartenance d’une zone à un site ou lister l’ensemble des parcelles d’une placette. Cette méthode ne permettait aucune évolutivité et imposait des logiques différentes incohérentes. Nous avons donc décidé d’adopter une solution plus classique consistant à créer de nouvelles tables, ce qui permettra à l’avenir une meilleure évolutivité et un accès plus simple aux appartenances de chaque élément (site, zone, placette, parcelle et plant).

Cette nouvelle solution consiste en l’ajout de l’ensemble d’entités et de relations suivant :



La standardisation

Le rendu précédent des sources ne présentait pas de ligne directrice dans l'application des conventions de nommage ni des règles de bonnes pratiques dans le code. Il a donc été nécessaire d'adopter une convention unique et claire. Nous avons décidé de nous conformer au *Standard de programmation SQL de niveau 1* proposé par **Samuel Dussault, Marc Dupuis, Christina Khnaisser et Luc Lavoie.**

Nous avons ainsi modifié l’ensemble des noms de types, d’attributs, de tables et de domaines afin d’assurer une cohérence globale dans la base de données.

Justification des changements dans la partie météo

Modifications des tables

Contexte

L'ancien modèle mariait chaque type d'observation (température, humidité, vents, pression, précipitations) à sa propre table (ObsTemperature, ObsHumidite, ObsVents, ...). Ce choix conduit à du "schéma par mesure" : chaque nouveau type de mesure nécessite une modification de la base de données en créant une nouvelle table. Cela complexifie inutilement l'ELT et augmente le risque d'incohérences.

Nous avons donc opté pour une solution où l’on réunirait dans une table centrale les différents types d’observation excepté les précipitations qui possèdent des types de données différentes du reste. En effet, pour température, humidité, vent etc. Il s’agit de valeurs minimales et maximales. Nous avons factorisé ces tables en une seule,

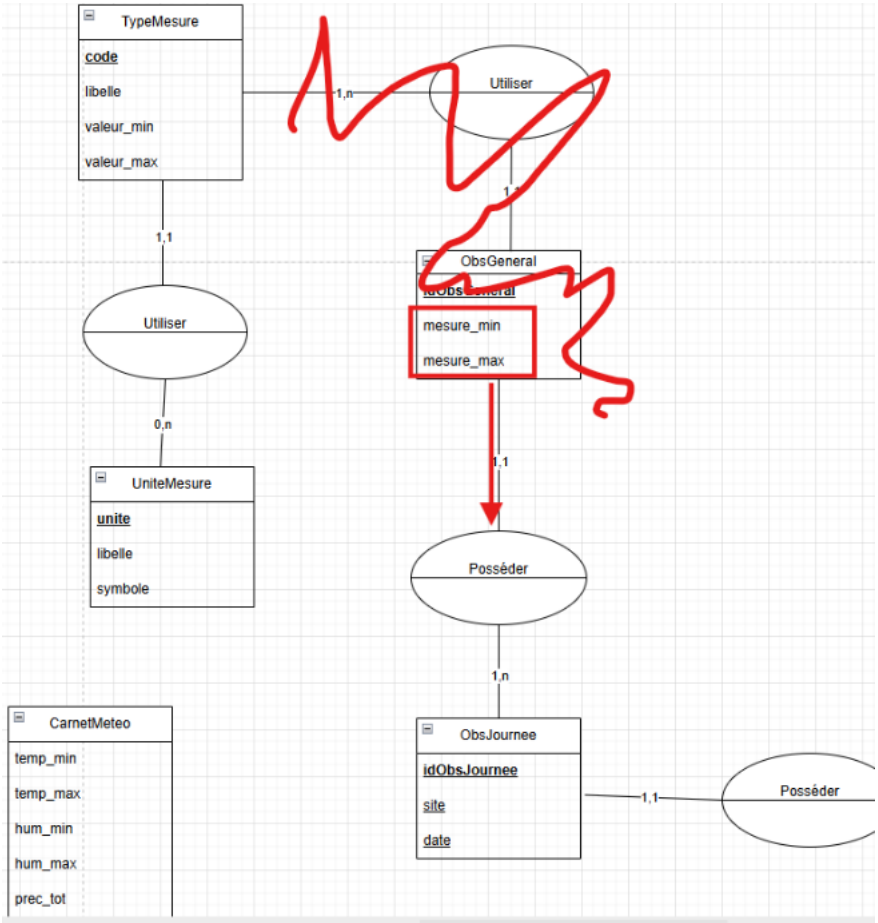
*Remplacement des tables « une par type » par un catalogue + table d'association (Type_mesure + Journee_mesure_observation)

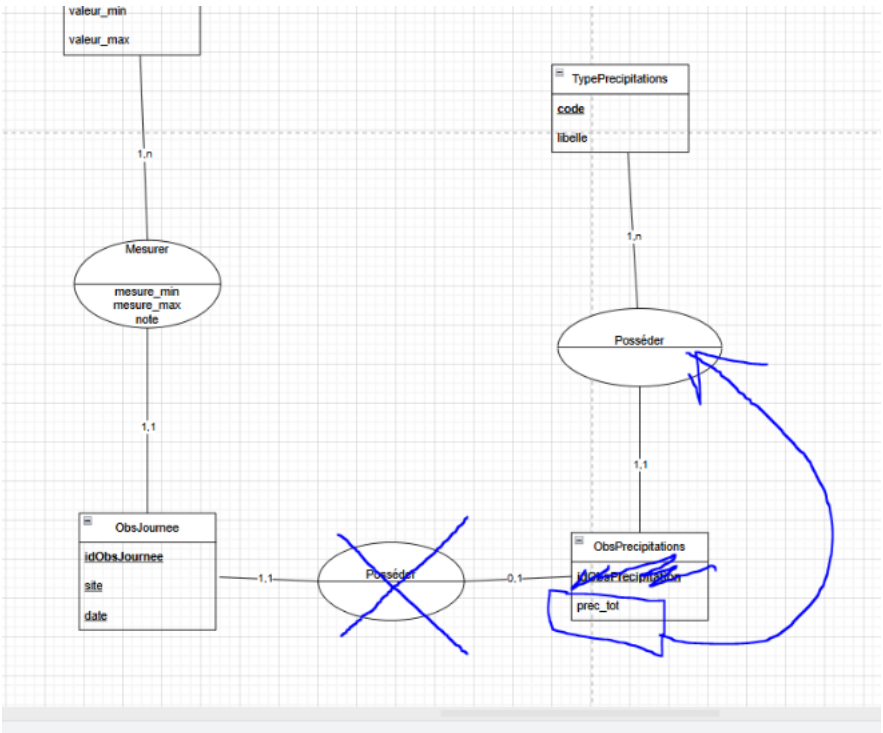
*Extensibilité : pour ajouter un nouveau type de mesure on insère un code dans Type_mesure au lieu d'ajouter une nouvelle table.

*Uniformité : toutes les mesures partagent la même sémantique (min/max, note) et peuvent être traitées par la même logique d'ELT/fonctions. Nous n’avons plus besoin de créer des fonctions spécifiques pour chaque type d’observations. Cela réduit donc fortement le temps de création de nos interfaces machines pour la suite du projet.

*Maintenance réduite : moins de scripts, moins d'indexes et moins de tests à maintenir.

Utilisation de clé composites





Dans le cadre du projet il nous a semblé important d'utiliser des tables d'association avec clé composite plutôt que de tables distinctes qui n'apportent pas de vraie valeur ajoutée avec des identifiants artificiels qui ne portent de sens sémantique. Avec les clés composites, il est plus facile de savoir de quoi on parle directement.

*Clarté sémantique : la clé composite (journee_observation_id, code_type_mesure) exprime directement l'unicité " type X mesuré pour la journée Y". De même, la clé composite (journee_observation_id, code_precipitation) exprime également l'unicité "type de précipitation X mesuré pour la journée Y". 3

*Évite la prolifération d'Ids artificiels sans signification métier.

*Contrainte d'intégrité naturelle : la contrainte de clé primaire composite empêche les doublons logiques.

Introduction des unités de mesure

Nous avons introduit le concept d'unité de mesure et de type d'unité afin d'y concentrer directement les unités ainsi que leur extrema.

*Normalisation des unités (Unite_mesure) et liaison Type_mesure → Unite_mesure

*Permet de stocker une seule fois le libellé, le symbole et les bornes (valeur_min/valeur_max) pour chaque unité.

*L'ELT a été modifier pour valider automatiquement les conversions et rejeter les mesures hors bornes.

Schéma relationnel actuel

Schéma relationnel herbivorie modifié

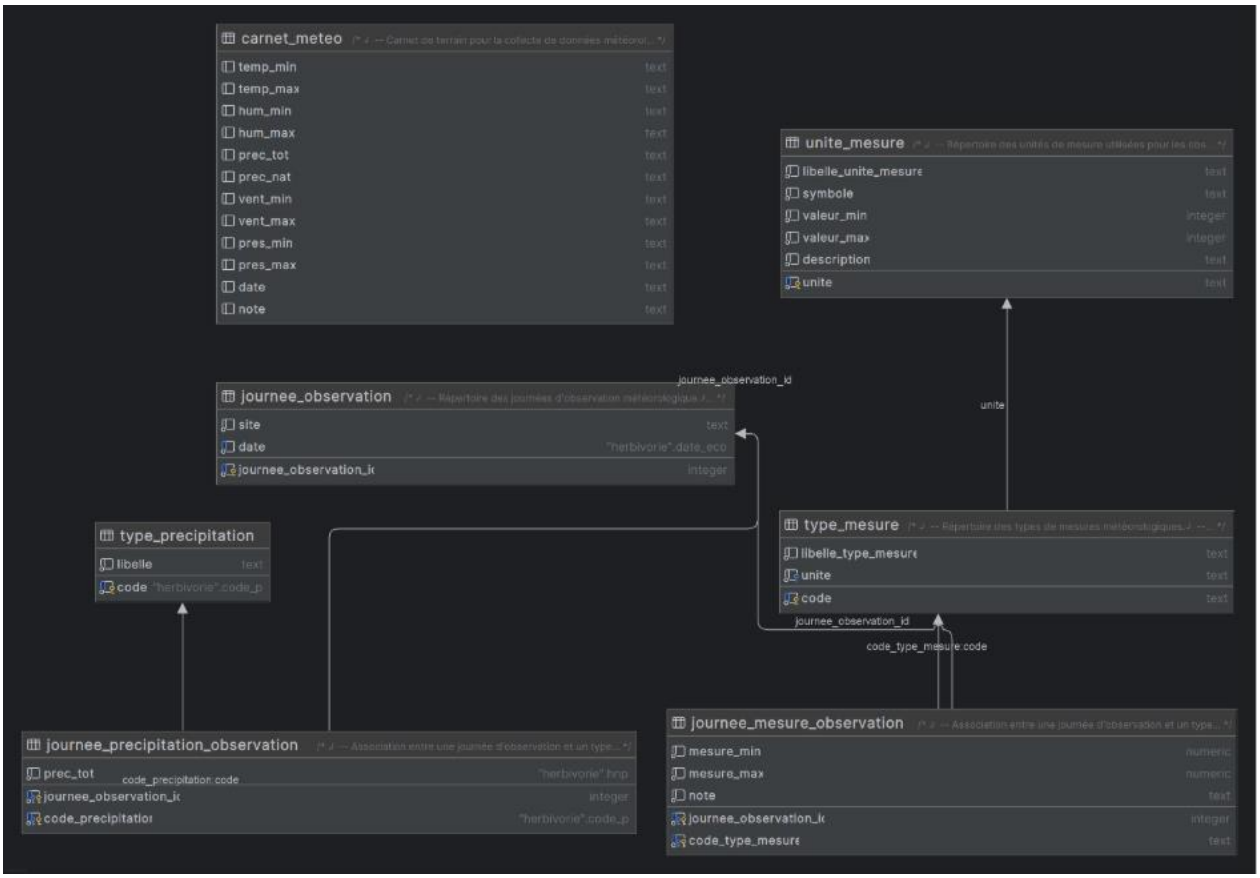


Schéma relationnel météo modifié

Précision et contraintes sur les types des attributs

Date_eco

Le type Date_eco représente une date utilisée pour consigner une observation écologique. Il impose que toute date enregistrée soit postérieure au 20 décembre 1582, ce qui correspond à l'adoption du calendrier grégorien.

```
CREATE DOMAIN Date_eco
DATE
CHECK (VALUE >= '1582-12-20');
```

Site_id

Site_id est un identifiant textuel composé exactement de deux lettres majuscules. Chaque site d'étude correspond à un lieu géographique particulier, et ce code assure une désignation courte et normalisée.

```
CREATE DOMAIN Site_id
TEXT
CHECK (VALUE ~ '^[A-Z]{2}$');
```

DescriptionDescription

Permet de stocker une note ou annotation liée à une observation ou à une entité du schéma. Ce texte doit contenir entre 1 et 60 caractères, ce qui garantit une description concise et contrôlée.

```
CREATE DOMAIN Description
TEXT
CHECK (CHAR_LENGTH(VALUE) BETWEEN 1 AND 60);
```

Zone_nom

Zone_nom représente un identifiant composé d'une ou plusieurs lettres majuscules. Il permet de distinguer les zones internes d'un site (ex. A, BC, AZC).

```
CREATE DOMAIN Zone_nom
TEXT
CHECK (VALUE ~ '^[A-Z]+$');
```

Placette_nom

Le type Placette_nom impose un format simple constitué d'une lettre majuscule suivie d'un chiffre. Il permet de désigner de manière standardisée chaque placette (ex. A1, B3).

```
CREATE DOMAIN Placette_nom
TEXT
CHECK (VALUE SIMILAR TO '[A-Z][0-9]');
```

Parcelle_nom

Ce type définit un code composé exactement de deux chiffres. Il permet de nommer les parcelles situées à l'intérieur des placettes.

```
CREATE DOMAIN Parcelle_nom
TEXT
CHECK (VALUE ~ '^[0-9]{2}$');
```

Arbre_id

Arbre_id permet d'identifier une espèce ou variété d'arbre à l'aide d'un texte court (1 à 20 caractères). Ce champ sert de clé de référence dans les descriptions forestières des placettes.

```
CREATE DOMAIN Arbre_id
```

```
TEXT
CHECK (CHAR_LENGTH(VALUE) BETWEEN 1 AND 20);
```

Peuplement_id

Ce type est un identifiant composé de quatre lettres majuscules. Il sert à catégoriser la nature du peuplement végétal d'une parcelle.

```
CREATE DOMAIN Peuplement_id
TEXT
CHECK (VALUE SIMILAR TO '[A-Z]{4}');
```

Hauteur_obstruction_id

Ce type entier permet d'exprimer une hauteur d'obstruction comprise entre 1 et 20 unités (échelle définit par les écologistes).

```
CREATE DOMAIN Hauteur_obstruction_id
INTEGER
CHECK (VALUE BETWEEN 1 AND 20);
```

Obstruction_id

Ce type textuel (1 à 2 caractères) sert à identifier les différents types possibles d'obstruction latérale dans une placette.

```
CREATE DOMAIN Obstruction_id
TEXT
CHECK (CHAR_LENGTH(VALUE) BETWEEN 1 AND 2);
```

Taux_val

Taux_val est un entier permettant de représenter un pourcentage, compris entre 0 et 100. Ce type est utilisé pour exprimer des taux d'obstruction ou de végétation au sol.

```
CREATE DOMAIN Taux_val
INTEGER
CHECK (VALUE BETWEEN 0 AND 100);
```

Rang_placette_arbre

Ce type entier représente le rang d'un arbre dominant, variant de 1 à 3. Il est employé pour déterminer la hiérarchie des arbres dominants d'une placette.

```
CREATE DOMAIN Rang_placette_arbre
INTEGER
CHECK (VALUE BETWEEN 1 AND 3);
```

type_vegetation_id

Le type permet d'encoder un identifiant textuel court (1 à 10 caractères) représentant une catégorie de végétation présente au sol.

```
CREATE DOMAIN type_vegetation_id
TEXT
CHECK (CHAR_LENGTH(VALUE) BETWEEN 1 AND 10);
```

Dim_mm

Ce type entier représente une dimension en millimètres, comprise entre 1 et 999. Il sert à mesurer la largeur ou la longueur des feuilles de trille lors des observations.

```
CREATE DOMAIN Dim_mm
INTEGER
CHECK (VALUE BETWEEN 1 AND 999);
```

Etat_id

Ce type identifie l'état d'un plant à l'aide d'une seule lettre majuscule. Il permet de codifier des états phénologiques ou morphologiques simples.

```
CREATE DOMAIN Etat_id
TEXT
CHECK (VALUE SIMILAR TO '[A-Z]{1}');
```

Temperature

Le type Temperature représente une mesure de la température de l'air ambiant, exprimée en degrés Celsius. Il impose que toute valeur enregistrée demeure dans l'intervalle des températures plausibles observables dans le sud du Québec. Cela garantit la cohérence des données météorologiques et prévient l'insertion de valeurs aberrantes.

```
CREATE DOMAIN Temperature
SMALLINT
CHECK (VALUE BETWEEN -50 AND 50);
```

Humidite

Le type Humidite permet de consigner un taux d'humidité absolue exprimé en pourcentage. Il couvre la plage complète des valeurs physiques possibles, allant de 0 % (air totalement sec) à 100 % (saturation complète). Cette contrainte assure la validité des données provenant de capteurs ou de relevés manuels.

```
CREATE DOMAIN Humidite
SMALLINT
CHECK (VALUE BETWEEN 0 AND 100);
```

Vitesse

Le type Vitesse représente la vitesse du vent mesurée en kilomètres par heure.Il impose un intervalle de valeurs allant de 0 à 300 km/h, couvrant toutes les situations météorologiques plausibles pour le contexte étudié, incluant vents calmes, modérés ou exceptionnellement violents.

```
CREATE DOMAIN Vitesse
SMALLINT
CHECK (VALUE BETWEEN 0 AND 300);
```

Pression

Le type Pression permet de stocker une mesure atmosphérique en hectopascals (hPa). Il limite les valeurs entre 900 et 1100 hPa, correspondant aux conditions normales observées au niveau du sol. Cette validation prévient l'insertion de données incohérentes ou corrompues.

```
CREATE DOMAIN Pression
SMALLINT
CHECK (VALUE BETWEEN 900 AND 1100);
```

HNP

Le type HNP (Hauteur Normée de Précipitations) permet de consigner une quantité de précipitations en millimètres. La valeur doit se situer entre 0 et 500 mm, ce qui couvre les intensités de précipitations possibles au cours d'une journée, qu'il s'agisse de bruines légères ou d'événements exceptionnels.

```
CREATE DOMAIN HNP
SMALLINT
CHECK (VALUE BETWEEN 0 AND 500);
```

Code_P

Le type Code_P sert à identifier la nature d'une précipitation à l'aide d'un code court, composé d'un seul caractère alphabétique majuscule. Ce format standardisé facilite le regroupement et la codification des différentes familles de précipitations (neige, pluie, grêle, etc.).

```
CREATE DOMAIN Code_P
CHAR(1)
CHECK (VALUE BETWEEN 'A' AND 'Z');
```

Interface Machine-Machine

Nous allons ici, afin de rendre l'exploitation de notre base de données plus fluide mettre en avant l'ensemble de nos fonction et procédure EMIR et ELT via un système indexé comme pour les types.

Lieux (Sites, Zones, Placettes, Parcelles)

Sites

Site_EVA()
Entrée : aucune
Sortie : TABLE(site_id Site_id, site_nom TEXT, site_description Description)
Description : Retourne la liste complète des sites d'étude.

Site_EVA_par_id(_site_id Site_id)
Entrée : _site_id (Site_id)
Sortie : TABLE(site_id, site_nom, site_description)
Description : Retourne un site correspondant à l'identifiant fourni.

Site_INS_S(_site_id Site_id, _site_nom TEXT, _site_description Description)
Entrée : identifiant, nom, description
Sortie : VOID
Description : Insère un nouveau site d'étude.

Site_MOD_S(_site_id Site_id, _site_nom TEXT, _site_description Description)
Entrée : identifiant du site + nouveaux champs
Sortie : VOID
Description : Met à jour toutes les informations d'un site.

Site_MOD_nom(_site_id Site_id, _site_nom TEXT)
Entrée : identifiant + nouveau nom
Sortie : VOID
Description : Met à jour uniquement le nom du site.

Site_MOD_description(_site_id Site_id, _site_description Description)
Entrée : identifiant + nouvelle description
Sortie : VOID
Description : Met à jour uniquement la description d'un site.

Site_RET_N(_site_id Site_id)

Entrée : identifiant
Sortie : VOID
Description : Supprime un site et ses dépendances.

Zones

Zone_EVA()
Entrée : aucune
Sortie : TABLE(zone_id, zone_nom, zone_description, site_id)
Description : Retourne la liste de toutes les zones.

Zone_EVA_par_id(_zone_id INTEGER)
Entrée : _zone_id
Sortie : TABLE(zone_id, zone_nom, zone_description, site_id)
Description : Retourne la zone correspondante.

Zone_INS_S(_zone_nom Zone_nom, _zone_description Description, _site_id Site_id)
Entrée : nom, description, site associé
Sortie : VOID
Description : Insère une nouvelle zone.

Zone_MOD_S(_zone_id INTEGER, _zone_nom Zone_nom, _zone_description Description, _site_id Site_id)
Entrée : identifiant + nouveaux champs
Sortie : VOID
Description : Met à jour toutes les informations d'une zone.

Zone_MOD_nom(_zone_id INTEGER, _zone_nom Zone_nom)
Entrée : identifiant + nouveau nom
Sortie : VOID
Description : Modifie uniquement le nom d'une zone.

Zone_MOD_description(_zone_id INTEGER, _zone_description Description)
Entrée : identifiant + nouvelle description
Sortie : VOID
Description : Modifie uniquement la description.

Zone_MOD_site_id(_zone_id INTEGER, _site_id Site_id)

Entrée : identifiant + nouveau site associé
Sortie : VOID
Description : Change le site auquel la zone appartient.

Zone_RET_N(_zone_id INTEGER)
Entrée : identifiant
Sortie : VOID
Description : Supprime une zone et ses dépendances.

Placettes

Placette_EVA()
Entrée : aucune
Sortie : TABLE(placette_id, placette_nom, placette_date, zone_id)
Description : Retourne toutes les placettes.

Placette_EVA_par_id(_placette_id INTEGER)
Entrée : identifiant
Sortie : TABLE(placette_id, placette_nom, placette_date, zone_id)
Description : Retourne une placette spécifique.

Placette_INS_S(_placette_nom Placette_nom, _placette_date Date_eco, _zone_id INTEGER)
Entrée : nom, date, zone associée
Sortie : VOID
Description : Insère une placette.

Placette100Parcelles_INS_S(_placette_id INTEGER)
Entrée : identifiant de la placette
Sortie : VOID
Description : Génère automatiquement 100 parcelles pour une placette.

Placette_MOD_S(_placette_id, _placette_nom, _placette_date, _zone_id)
Entrée : identifiant + nouveaux champs
Sortie : VOID
Description : Met à jour toutes les informations d'une placette.

Placette_MOD_nom(_placette_id, _placette_nom)

Entrée : identifiant + nouveau nom
Sortie : VOID
Description : Modifie le nom.

Placette_MOD_date(_placette_id, _placette_date)
Entrée : identifiant + nouvelle date
Sortie : VOID
Description : Modifie la date.

Placette_MOD_zone_id(_placette_id, _zone_id)
Entrée : identifiant + nouvelle zone
Sortie : VOID
Description : Change l'appartenance à une zone.

Placette_RET_N(_placette_id)
Entrée : identifiant
Sortie : VOID
Description : Supprime une placette et ses dépendances.

Parcelles

Parcelle_EVA()
Entrée : aucune
Sortie : TABLE(parcelle_id, parcelle_nom, parcelle_description, placette_id)
Description : Retourne toutes les parcelles.

Parcelle_EVA_par_id(_parcelle_id)
Entrée : identifiant
Sortie : TABLE(parcelle_id, parcelle_nom, parcelle_description, placette_id)
Description : Retourne une parcelle spécifique.

Parcelle_INS_S(_parcelle_nom, _parcelle_description, _placette_id)
Entrée : nom, description, placette associée
Sortie : VOID
Description : Insère une parcelle.

Parcelle_MOD_S(_parcelle_id, _parcelle_nom, _parcelle_description, _placette_id)
Entrée : identifiant + nouvelles valeurs

Sortie : VOID
Description : Met à jour une parcelle complète.

Parcelle_MOD_nom(_parcelle_id, _parcelle_nom)
Entrée : identifiant + nouveau nom
Sortie : VOID
Description : Modifie le nom.

Parcelle_MOD_description(_parcelle_id, _parcelle_description)
Entrée : identifiant + description
Sortie : VOID
Description : Modifie la description.

Parcelle_MOD_placette_id(_parcelle_id, _placette_id)
Entrée : identifiant + nouvelle placette
Sortie : VOID
Description : Change la placette associée.

Parcelle_RET_N(_parcelle_id)
Entrée : identifiant
Sortie : VOID
Description : Supprime une parcelle.

Caractérisation des placettes

Arbre

Arbre_EVA()
Entrée : aucune
Sortie : TABLE(arbre_id Arbre_id, arbre_description Description)
Description : Retourne la liste de tous les arbres.

Arbre_EVA_par_id(_arbre_id Arbre_id)
Entrée : _arbre_id
Sortie : TABLE(arbre_id, arbre_description)
Description : Retourne un arbre selon son identifiant.

Peuplement

Peuplement_EVA()
Entrée : aucune
Sortie : TABLE(peuplement_id Peuplement_id, peuplement_description Description)
Description : Retourne tous les peuplements.

Peuplement_EVA_par_id(_peuplement_id Peuplement_id)
Entrée : identifiant du peuplement
Sortie : TABLE(peuplement_id, peuplement_description)
Description : Retourne un peuplement spécifique.

Obstruction_hauteur

Obstruction_hauteur_EVA()
Entrée : aucune
Sortie : TABLE(obstruction_hauteur Hauteur_obstruction_id)
Description : Retourne toutes les hauteurs d'obstruction.

Obstruction_hauteur_EVA_par_id(_obstruction_hauteur Hauteur_obstruction_id)
Entrée : hauteur d'obstruction
Sortie : TABLE(obstruction_hauteur)
Description : Retourne une hauteur d'obstruction spécifique.

Obstruction_type

Obstruction_type_EVA()
Entrée : aucune
Sortie : TABLE(obstruction_type_id Obstruction_id, obstruction_type_description Description)
Description : Retourne tous les types d'obstruction.

Obstruction_type_EVA_par_id(_obstruction_type_id Obstruction_id)
Entrée : identifiant du type
Sortie : TABLE(obstruction_type_id, obstruction_type_description)
Description : Retourne un type d'obstruction spécifique.

Rang_arbre

Rang_arbre_EVA()
Entrée : aucune
Sortie : TABLE(rang_arbre_id Rang_placette_arbre, rang_arbre_description Description)
Description : Retourne tous les rangs d'arbres.

Rang_arbre_EVA_par_id(_rang_arbre_id Rang_placette_arbre)
Entrée : identifiant
Sortie : TABLE(rang_arbre_id, rang_arbre_description)
Description : Retourne un rang d'arbre spécifique.

Type_vegetation

Type_vegetation_EVA()
Entrée : aucune
Sortie : TABLE(type_vegetation_id TEXT, type_vegetation_description Description)
Description : Retourne tous les types de végétation.

Type_vegetation_EVA_par_id(_type_vegetation_id TEXT)
Entrée : identifiant textuel
Sortie : TABLE(type_vegetation_id, type_vegetation_description)
Description : Retourne un type de végétation spécifique.

Placette_obstruction

Placette_obstruction_EVA()
Entrée : aucune
Sortie :
TABLE(
placette_id INTEGER,
obstruction_type_id Obstruction_id,
obstruction_hauteur Hauteur_obstruction_id,
placette_obstruction_taux Taux_val,
placette_obstruction_date Date_eco
)
Description : Retourne toutes les obstructions relevées sur les placettes.

Placette_obstruction_EVA_par_placette_id(_placette_id INTEGER)
Entrée : identifiant de la placette

Sortie : même structure que ci-dessus
Description : Retourne les obstructions associées à une placette.

Placette_peuplement

Placette_peuplement_EVA()
Entrée : aucune
Sortie : TABLE(placette_id, peuplement_id, placette_peuplement_date)
Description : Retourne les peuplements enregistrés pour les placettes.

Placette_peuplement_EVA_par_placette_id(_placette_id INTEGER)
Entrée : placette_id
Sortie : TABLE(placette_id, peuplement_id, placette_peuplement_date)
Description : Retourne tous les peuplements d'une placette.

Placette_arbre

Placette_arbre_EVA()
Entrée : aucune
Sortie : TABLE(placette_id, arbre_id, rang_id, placette_arbre_date)
Description : Retourne tous les arbres associés aux placettes.

Placette_arbre_EVA_par_placette_id(_placette_id INTEGER)
Entrée : placette_id
Sortie : TABLE(placette_id, arbre_id, rang_id, placette_arbre_date)
Description : Retourne les arbres d'une placette.

Placette_occupation

Placette_occupation_EVA()
Entrée : aucune
Sortie : TABLE(placette_id, type_vegetation_id, placette_occupation_taux, placette_occupation_date)
Description : Retourne les types de végétation observés dans les placettes.

Placette_occupation_EVA_par_placette_id(_placette_id INTEGER)
Entrée : identifiant de placette
Sortie : même structure que ci-dessus

Description : Retourne les observations d'occupation pour une placette donnée.

Plants et observations

Plant

Plant_EVA()
Entrée : aucune
Sortie :
TABLE(
plant_id INTEGER,
plant_date Date_eco,
plant_note TEXT,
parcelle_id INTEGER
)
Description : Retourne la liste complète des plants enregistrés.

Plant_EVA_par_id(_plant_id INTEGER)
Entrée : _plant_id
Sortie : TABLE(plant_id, plant_date, plant_note, parcelle_id)
Description : Retourne les informations du plant correspondant à l'ID fourni.

Plant_EVA_par_parcelle_id(_parcelle_id INTEGER)
Entrée : identifiant de parcelle
Sortie : TABLE(plant_id, plant_date, plant_note, parcelle_id)
Description : Retourne tous les plants appartenant à une parcelle donnée.

Obs_dimension

Obs_dimension_EVA()
Entrée : aucune
Sortie :
TABLE(
plant_id INTEGER,
obs_dimension_longueur Dim_mm,
obs_dimension_largeur Dim_mm,
obs_dimension_date Date_eco,
obs_dimension_note TEXT

)
Description : Retourne toutes les observations de dimensions (longueur / largeur) des plants.

Obs_dimension_EVA_par_plant_id(_plant_id INTEGER)
Entrée : identifiant du plant
Sortie : même structure que ci-dessus
Description : Retourne toutes les mesures dimensionnelles d'un plant.

Obs_dimension_EVA_par_pk(_plant_id INTEGER, _obs_dimension_date Date_eco)
Entrée :_plant_id , _obs_dimension_date
Sortie : TABLE(plant_id, obs_dimension_longueur, obs_dimension_largeur, obs_dimension_date, obs_dimension_note)
Description : Retourne une observation de dimension identifiée par sa clé primaire (plant_id + date).

Obs_floraison

Obs_floraison_EVA()
Entrée : aucune
Sortie :
TABLE(
plant_id INTEGER,
obs_floraison_date Date_eco,
obs_floraison_note TEXT
)
Description : Retourne toutes les observations de floraison des plants.

Obs_floraison_EVA_par_plant_id(_plant_id INTEGER)
Entrée : identifiant du plant
Sortie : TABLE(plant_id, obs_floraison_date, obs_floraison_note)
Description : Retourne toutes les observations de floraison pour un plant donné.

Obs_floraison_EVA_par_pk(_plant_id INTEGER, _obs_floraison_date Date_eco)
Entrée :_plant_id, _obs_floraison_date
Sortie : TABLE(plant_id, obs_floraison_date, obs_floraison_note)
Description : Retourne une observation de floraison spécifique (clé primaire plant_id + date).

Etat (état d'un plant)

Etat_EVA()
Entrée : aucune
Sortie : TABLE(etat_id Etat_id, etat_description Description)
Description : Retourne tous les états possibles d'un plant (ex. sain, flétri, etc.).

Etat_EVA_par_id(_etat_id Etat_id)
Entrée : identifiant de l'état
Sortie : TABLE(etat_id, etat_description)
Description : Retourne un état particulier selon son identifiant.

Obs_etat

Obs_etat_EVA()
Entrée : aucune
Sortie :
TABLE(
plant_id INTEGER,
etat_id Etat_id,
obs_etat_date Date_eco,
obs_etat_note TEXT
)
Description : Retourne toutes les observations d'état des plants.

Obs_etat_EVA_par_plant_id(_plant_id INTEGER)
Entrée : identifiant du plant
Sortie : TABLE(plant_id, etat_id, obs_etat_date, obs_etat_note)
Description : Retourne toutes les observations d'état d'un plant.

Obs_etat_EVA_par_pk(_plant_id INTEGER, _obs_etat_date Date_eco)
Entrée : _plant_id , _obs_etat_date
Sortie : TABLE(plant_id, etat_id, obs_etat_date, obs_etat_note)
Description : Retourne une observation d'état identifiée par sa clé primaire (plant_id + date).

Météo

Journées d'observation

Meteo_consulter_journee(p_date Date_eco)

Entrée : p_date (Date_eco)
Sortie : TABLE(
journee_observation_id INTEGER,
site TEXT, date Date_eco,
temp_min NUMERIC,
temp_max NUMERIC,
hum_min NUMERIC,
hum_max NUMERIC,
vent_min NUMERIC,
vent_max NUMERIC,
pres_min NUMERIC,
pres_max NUMERIC,
prec_tot HNP,
code_precipitation Code_P,
note TEXT)
Description : Retourne toutes les informations d'une journée d'observation météorologique pour une date donnée, incluant les mesures (température, humidité, vent, pression) et les précipitations.

Meteo_consulter_periode(p_date_debut Date_eco, p_date_fin Date_eco)

Entrée : p_date_debut (Date_eco), p_date_fin (Date_eco)
Sortie : TABLE(
journee_observation_id INTEGER,
site TEXT,
date Date_eco,
temp_min NUMERIC,
temp_max NUMERIC,
hum_min NUMERIC,
hum_max NUMERIC,
vent_min NUMERIC,
vent_max NUMERIC,
pres_min NUMERIC,
pres_max NUMERIC,
prec_tot HNP,
code_precipitation Code_P,

note TEXT)
Description : Retourne toutes les observations météorologiques pour une période donnée, triées par date.

Meteo_obtenir_journee_id(p_date Date_eco)
Entrée : p_date (Date_eco)
Sortie : INTEGER
Description : Retourne l'identifiant d'une journée d'observation pour une date donnée.

Meteo_inserer_journee(p_site TEXT, p_date Date_eco)
Entrée : p_site (TEXT), p_date (Date_eco)
Sortie : INTEGER (journee_observation_id)
Description : Insère ou met à jour une journée d'observation. Retourne l'ID de la journée. Si la date existe déjà, met à jour le site.

Meteo_modifier_journee(p_journee_observation_id INTEGER, p_site TEXT, p_date Date_eco)
Entrée : p_journee_observation_id (INTEGER), p_site (TEXT, optionnel), p_date (Date_eco, optionnel)
Sortie : BOOLEAN
Description : Met à jour les informations d'une journée d'observation existante. Les paramètres optionnels permettent de modifier uniquement le site ou la date.

Meteo_supprimer_journee(p_journee_observation_id INTEGER)
Entrée : p_journee_observation_id (INTEGER)
Sortie : BOOLEAN
Description : Supprime une journée d'observation complète avec toutes ses mesures et précipitations.

Mesures météorologiques

Meteo_inserer_mesure(p_journee_observation_id INTEGER, p_code_type_mesure TEXT, p_mesure_min NUMERIC, p_mesure_max NUMERIC, p_note TEXT)
Entrée : p_journee_observation_id (INTEGER),
p_code_type_mesure (TEXT : TEMP, HUM, VENT, PRES),
p_mesure_min (NUMERIC),
p_mesure_max (NUMERIC),
p_note (TEXT, optionnel)
Sortie : BOOLEAN

Description : Insère ou met à jour une mesure météorologique pour une journée d'observation. Vérifie que la mesure minimale est inférieure ou égale à la mesure maximale.

Meteo_modifier_mesure(p_journee_observation_id INTEGER, p_code_type_mesure TEXT, p_mesure_min NUMERIC, p_mesure_max NUMERIC, p_note TEXT)
Entrée : p_journee_observation_id (INTEGER),
p_code_type_mesure (TEXT),
p_mesure_min (NUMERIC, optionnel),
p_mesure_max (NUMERIC, optionnel),
p_note (TEXT, optionnel)
Sortie : BOOLEAN
Description : Modifie une mesure météorologique existante. Les paramètres optionnels permettent de modifier uniquement certaines valeurs.

Meteo_supprimer_mesure(p_journee_observation_id INTEGER, p_code_type_mesure TEXT)
Entrée : p_journee_observation_id (INTEGER), p_code_type_mesure (TEXT)
Sortie : BOOLEAN
Description : Supprime une mesure météorologique spécifique.

Précipitations

Meteo_inserer_precipitation(p_journee_observation_id INTEGER, p_code_precipitation Code_P, p_prec_tot HNP)
Entrée : p_journee_observation_id (INTEGER), p_code_precipitation (Code_P : P, N),
p_prec_tot (HNP)
Sortie : BOOLEAN
Description : Insère ou met à jour une précipitation pour une journée d'observation.

Meteo_modifier_precipitation(p_journee_observation_id INTEGER, p_code_precipitation Code_P, p_prec_tot HNP)
Entrée : p_journee_observation_id (INTEGER),
p_code_precipitation (Code_P),
p_prec_tot (HNP, optionnel)
Sortie : BOOLEAN
Description : Modifie une précipitation existante.

Meteo_supprimer_precipitation(p_journee_observation_id INTEGER, p_code_precipitation Code_P)

Entrée : p_journee_observation_id (INTEGER), p_code_precipitation (Code_P)
Sortie : BOOLEAN
Description : Supprime une précipitation spécifique.

Vues et procédures spéciales

Meteo_conditions_sans_precipitation (Vue Y3)

Entrée : aucune (vue)
Sortie : TABLE(
journee_observation_id INTEGER,
site TEXT,
date Date_eco,
temp_min NUMERIC,
temp_max NUMERIC,
hum_min NUMERIC,
hum_max NUMERIC,
vent_min NUMERIC,
vent_max NUMERIC,
pres_min NUMERIC,
pres_max NUMERIC,
note TEXT)
Description : Vue donnant les conditions météorologiques complètes (température, humidité, vent, pression) mais excluant les précipitations.

Meteo_retirer_periode_temperature(p_date_debut Date_eco, p_date_fin Date_eco, p_temperature_seuil NUMERIC) (Procédure Y4)

Entrée : p_date_debut (Date_eco), p_date_fin (Date_eco), p_temperature_seuil (NUMERIC)
Sortie : VOID
Description : Retire les données météorologiques pour une période donnée si la température minimale rapportée est en deçà d'une température donnée. Supprime toutes les journées d'observation dont la température minimale est inférieure au seuil.

Meteo_augmenter_temperatures_periode(p_date_debut Date_eco, p_date_fin Date_eco, p_pourcentage NUMERIC) (Procédure Y5)

Entrée : p_date_debut (Date_eco), p_date_fin (Date_eco), p_pourcentage (NUMERIC)
Sortie : VOID

Description : Augmente les températures rapportées d'un pourcentage donné durant une période donnée. Le pourcentage doit être supérieur ou égal à -100. Si l'augmentation inverse l'ordre min/max, les valeurs sont automatiquement échangées.

ELT

Meteo_ELT()

Entrée : aucune

Sortie : VOID

Description : Procédure d'importation des données météorologiques depuis la table Carnet_meteo vers les tables Journee_observation, Journee_mesure_observation et Journee_precipitation_observation. Effectue la validation et la conversion des données selon les domaines définis (Date_eco, Temperature, Humidite, Vitesse, Pression, HNP, Code_P). Les données invalides sont ignorées (ON CONFLICT DO NOTHING).