CPI A1 2024-2025

ClearData

BLOC BDD
LIVRABLE N°2

Projet Données et modélisation

BORDRON MATHIS
DAVID SIMON
DAGNAS SWAN
GOURLAOUEN TANGUY

Table des matières

Contexte

Nous faisons partie de l'équipe « Data » de l'entreprise Clear Data. Nous sommes chargées de créer une base de données non redondante et cohérente comportant des relevés publics sur la qualité de l'air dans les grandes villes de France. Cette liste pourra être modifiée, consultée, ajoutée et supprimée facilement. Il nous est demandé notamment de lister l'ensemble des agences recueillant des informations ainsi que le personnel de ces agences. Il nous faudra aussi recenser l'ensemble des relevés à propos des différents GES (Gaz à Effet de Serre) et GESI (Gaz à Effet de Serre provenant de l'Industrie).

Pour ce livrable, il vous sera présenté les 12 requêtes en langage SQL demandées, ainsi qu'une explication détaillée du processus de génération et de peuplement de la base de données. Il fait également le point sur les fonctionnalités développées, en précisant celles qui sont conformes aux spécifications techniques et celles qui sont absentes ou incomplètes.

Rappel du Livrable 1

1) OBS

Le schéma OBS est une représentation hiérarchique qui expose visuellement la responsabilité de chaque personne dans le projet. Pour ce faire, nous avons commencé par reprendre le schéma réalisé dans le Livrable 0 afin de le compléter.

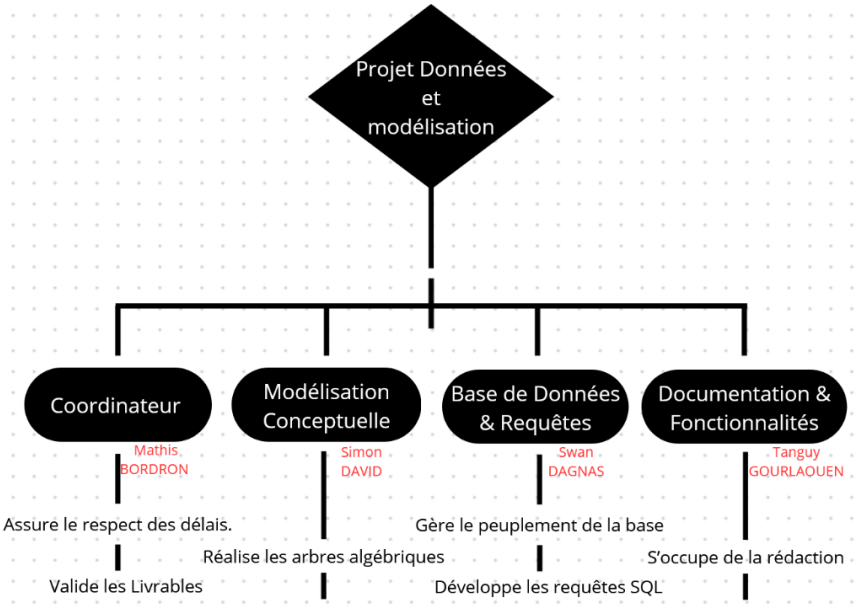


Figure 1: OBS

2) WBS

Le schéma WBS est une représentation hiérarchique qui expose visuellement toutes les étapes nécessaires afin de réaliser les différentes tâches d'un projet. Dans notre cas, il nous permet de démontrer en sous-étapes l'organisations des livrables concernés

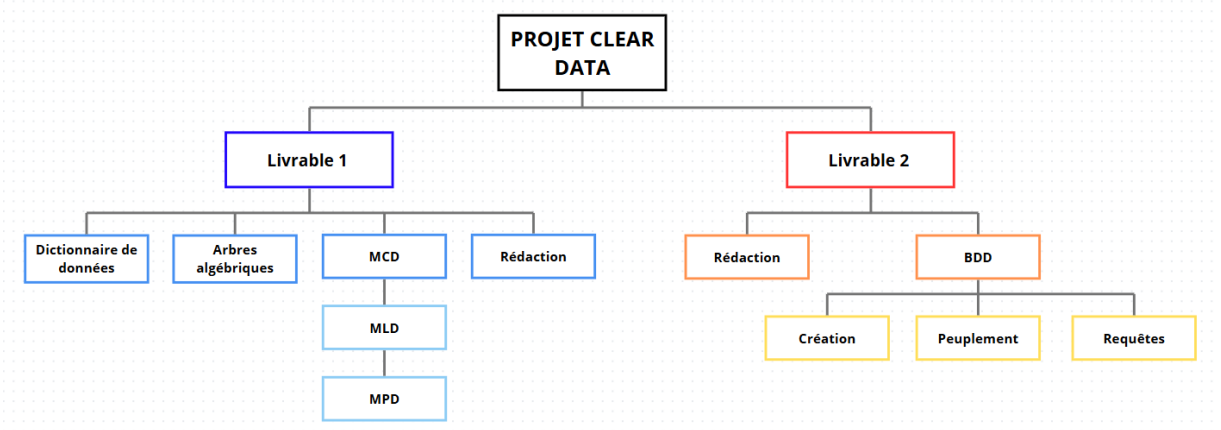


Figure 2: WBS

3) MLD

Nous avons trouvé nécessaire de réafficher le MLD qui était présent dans notre Livrable 1 il nous permet de ne pas oublier les différentes relations entre les tables. Cela nous permet aussi de mieux comprendre la réalisation des différentes requêtes de 8 à 12 qui n'ont pas eu d'arbre algébrique dans le Livrable 1.

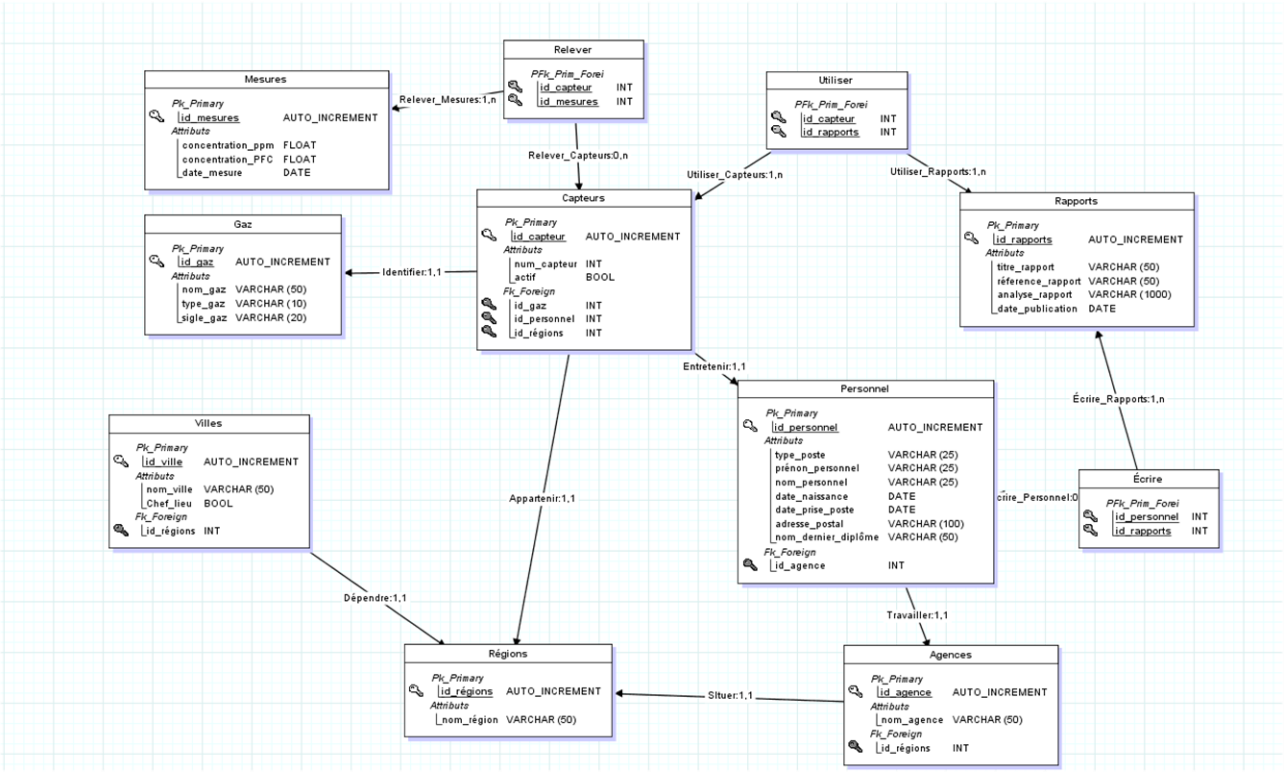


Figure 3: MLD

Requêtes en langage SQL demandées

1) Liste de l'ensemble des agences :

Pour réaliser cette première requête, nous sommes partis de l'arbre algébrique effectué lors du livrable 1. Comme nous pouvons le voir, il s'agit de la rédaction de la plus courte possible. Nous n'utilisons qu'un SELECT et un FROM.

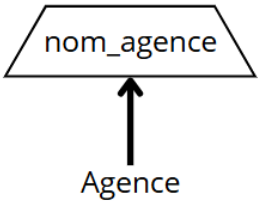


Figure 5 : Arbre Algébrique Requête 1

```
SELECT Nom_agence
FROM agences
ORDER BY Nom_agence;
```

Figure 6: Code Requête 1

	Nom_agence
1	Agence_Ajaccio
2	Agence_Bordeaux
3	Agence_Dijon
4	Agence_Lille
5	Agence_Limoges
6	Agence_Lyon
7	Agence_Marseille
8	Agence_Nantes
9	Agence_Orléans
10	Agence_Paris
11	Agence_Rennes
12	Agence_Rouen
13	Agence_Strasbourg
14	Agence_Toulouse

Figure 4: Résultat Requête 1

2) Liste de l'ensemble du personnel technique de l'agence de Bordeaux :

Cette seconde requêtes nécessite une petite modification supplémentaire. La rédaction est toujours la même or nous avons besoin d'effectuer une jonction pour accéder à l'ensemble des informations que nous souhaitons afficher. Cette jonction s'effectue dans la partie FROM entre les tables Personnel et Agence.

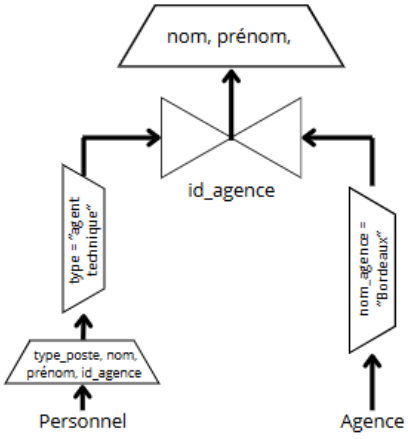


Figure 7: Arbre Algébrique Requête 2

```sql
SELECT Nom_personnel, Prénon_personnel, Type_poste, Nom_agence
FROM agences
Join personnels p on agences.ID_AGENCE = p.ID_AGENCE
WHERE Type_poste = 'agent technique' AND Nom_agence = 'Agence_Bordeaux';
```

Figure 8: Code Requête 2

	Nom_personnel	Prénon_personnel	Type_poste	Nom_agence
1	Julien	Guérin	agent technique	Agence_Bordeaux
2	Emile	Bonnin	agent technique	Agence_Bordeaux
3	Julie	Goncalves	agent technique	Agence_Bordeaux
4	Robert	Lévêque	agent technique	Agence_Bordeaux
5	Albert	Noël	agent technique	Agence_Bordeaux
6	Victoire	Paris	agent technique	Agence_Bordeaux
7	Henry	Aubert	agent technique	Agence_Bordeaux
8	Gilles	Lebreton	agent technique	Agence_Bordeaux

Figure 9: Résultat Requête 2

3) *Nombre total de capteurs déployés :*

Dans cette troisième requête nous allons utiliser une nouvelle fonction qui est un agrégat pour ce qui est du code cette fonction s'exécute dans la partie SELECT. Dans cette requête il s'agit du la formule « COUNT » qui sera accompagné de l'ID capteur c'est ce que nous allons donc compter. Pour ce qui est du reste la rédaction ne change pas.

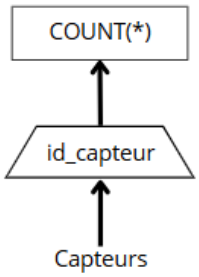


Figure 11 : Arbre Algébrique Requête 3

```sql
SELECT COUNT(*) as Nombre_capteurs_déployés
FROM capteurs;
```

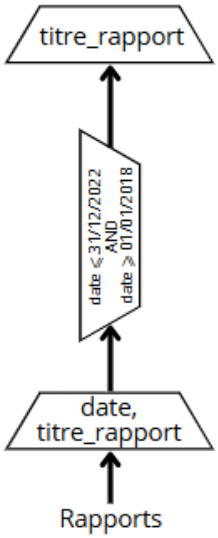
Figure 10: Code Requête 3

	Nombre_capteurs_déployés
1	576

Figure 12: Résultat Requête 3

4) Liste des rapports publiés entre 2018 et 2022 :

Cette quatrième requête n'utilise aucune nouvelle fonction comme la 2nd requête nous allons effectuer un filtre ne garder uniquement les rapports d'une certaine période. Cette contrainte va donc être insérer dans la partie WHERE, puisqu'il s'agit d'une période nous allons pouvoir utiliser le formule BETWEEN nous permettant d'avoir une plage de temps.




```sql
SELECT *
FROM rapports
WHERE YEAR(date_publication) BETWEEN 2018 AND 2022;
```

Figure 13: Code Requête 4

Figure 14 : Arbre Algébrique Requête 4

	ID_RAPPORT	Titre_rapport	Référence_rapport	Analyse_rapport
1	25	Gestion des déchets et transition écologique	BK-0158-BMZ	Le thème aborde les implications de politi
2	26	Approches systémiques de la transition écologique	DQ-5572-OQA	Ce rapport explore en profondeur impact de
3	27	Empreinte carbone des chaînes d'approvisionnement	ZC-3033-GJZ	Le thème aborde les implications de transi
4	28	Implication des jeunes dans l'action climatique	ZY-9467-IVR	Le sujet met en évidence les défis associé
5	29	Stratégies locales de résilience climatique	RG-7024-FOJ	Le sujet met en évidence les défis associé
6	30	Infrastructures vertes et durables	DG-9814-QUD	Le thème aborde les implications de gestio
7	31	Impacts climatiques sur la sécurité hydrique	VX-2004-NJG	Ce rapport explore en profondeur emissions

Figure 15: Résultat Requête 4

5) *Afficher les concentrations de CH4 (en ppm) dans les régions « Ile-de-France », « Bretagne » et « Occitanie » en mai et juin 2023 :*

Cette cinquième requête peut paraitre compliqué mais en réalité la réalisation du code n'est pas bien difficile, elle suit la même rédaction que les requêtes précédentes. On peut observer a l'aide de l'arbre algébrique que plusieurs jonctions sont à réaliser nous allons donc toute les réaliser en même temps dans la section « FROM ». Une fois ces jonctions faites nous avons donc accès à toutes les informations et donc effectuer l'ensemble de notre contrainte dans la partie « WHERE » à savoir les contraintes de gaz, période et lieu.

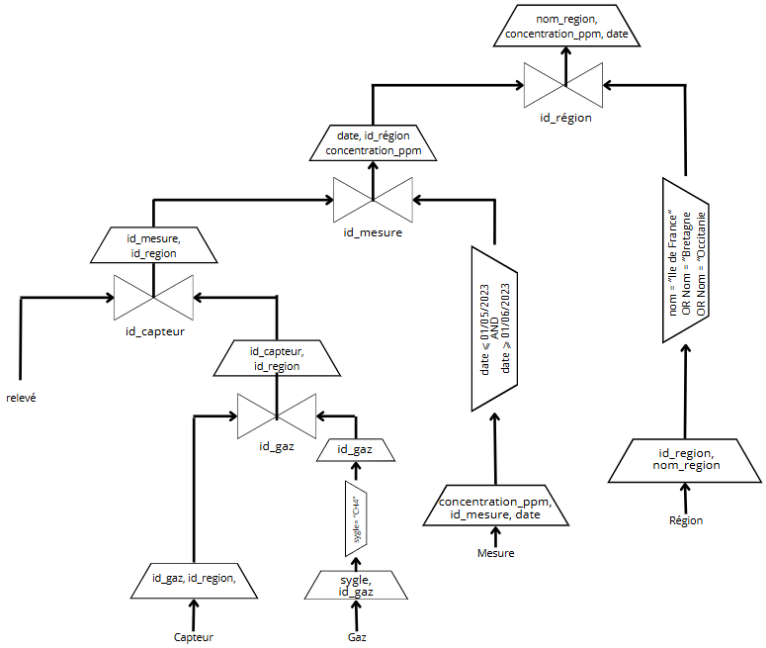


Figure 16: Arbre Algébrique Requête 5

```
SELECT DISTINCT mesures.Concentration_ppm, gaz.Sigle_gaz, régions.Nom_région, mesures.Date_mesure
FROM mesures
JOIN relever 1<->1..n: ON mesures.ID_MESURE = relever.ID_MESURE
JOIN capteurs 1..n<->1: ON relever.ID_CAPTEUR = capteurs.ID_CAPTEUR
JOIN gaz 1..n<->1: ON capteurs.ID_GAZ = gaz.ID_GAZ
JOIN régions 1..n<->1: ON capteurs.ID_RÉGION = régions.ID_RÉGION
WHERE mesures.Date_mesure BETWEEN '2023-05-01' AND '2023-06-30'
AND gaz.Sigle_gaz = 'CH4'
AND régions.Nom_région IN ('Île-de-France', 'Occitanie', 'Bretagne')
ORDER BY régions.Nom_région, mesures.Date_mesure;
```

Figure 17: Code Requête 5

	Concentration_ppm	Sigle_gaz	Nom_région	Date_mesure
1	1359.7	CH4	Bretagne	2023-05-01
2	986.15	CH4	Bretagne	2023-05-01
3	1337.13	CH4	Bretagne	2023-05-01
4	1328.78	CH4	Bretagne	2023-05-01
5	42.46	CH4	Bretagne	2023-06-01
6	85.64	CH4	Bretagne	2023-06-01

Figure 18: Résultat Requête 5

6) *Liste des noms des agents techniques maintenant des capteurs concernant les gaz à effet de serre provenant de l'industrie (GESI) :*

Pour cette sixième requête elle suit le même fonctionnement que la requête précédente à savoir plusieurs jonctions simultanées dans la section « FROM » et l'insertion de l'ensemble de nos contraintes dans la section « WHERE ». On peut voir que cette rédaction ne suit pas exactement le fil de notre arbre algébrique (on par exemple réaliser des sélection intermédiaire). L'avantage de ce code est qu'il est simple à réaliser mais aussi à comprendre cela réduit donc les potentiels différentes erreurs à différents endroits du code. En revanche le problème est que beaucoup de données vont être gardé alors qu'elles ne seront jamais sollicitées cela prend donc une grande place de stockage.

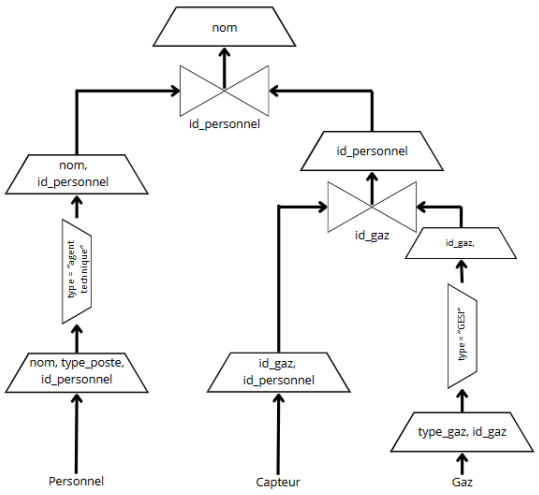


Figure 20: Arbre Algébrique Requête 6

```
SELECT DISTINCT Nom_personnel, Prénon_personnel, Type_gaz, Type_poste
FROM personnels
JOIN capteurs 1<->1..n: on personnels.ID_AGENT = capteurs.ID_AGENT
JOIN gaz 1..n<->1: on gaz.ID_GAZ = capteurs.ID_GAZ
WHERE Type_gaz = 'GESI' AND Type_poste = 'agent technique'
ORDER BY Nom_personnel;
```

Figure 21: Code Requête 6

	Nom_personnel	Prénon_personnel	Type_gaz	Type_poste
1	Alphonse	Prévost	GESI	agent technique
2	Andre	Georges	GESI	agent technique
3	Anna	Allain	GESI	agent technique
4	Anne	Paul	GESI	agent technique
5	Antoine	Perret	GESI	agent technique
6	Augustin	Guillet	GESI	agent technique
7	Catherine	Caron	GESI	agent technique
8	Elisabeth	Parent	GESI	agent technique

Figure 19: Résultat Requête 6

7) *Titres et dates des rapports concernant des concentrations de NH3, classés par ordre anti-chronologique :*

Pour cette septième requête nous allons encore une fois effectuer la même rédaction que les deux dernières. Une spécificité s'ajoute qui n'avait pas été effectué dans le livrable 1 il s'agit de la partie « ordre antichronologique » pour cela nous allons donc devoir ajouter une ligne a notre code nous permettant de ranger les valeurs dans l'ordre souhaité. Il s'agit de la fonction « ORDER BY » on y ajoute à la suite le nom de la colonne qui vas être classé et à la fin l'ordre c'est-à-dire croissant lorsque l'on met « ASC » et décroissant pour « DESC » c'est ce que nous avons utilisé dans notre situation.

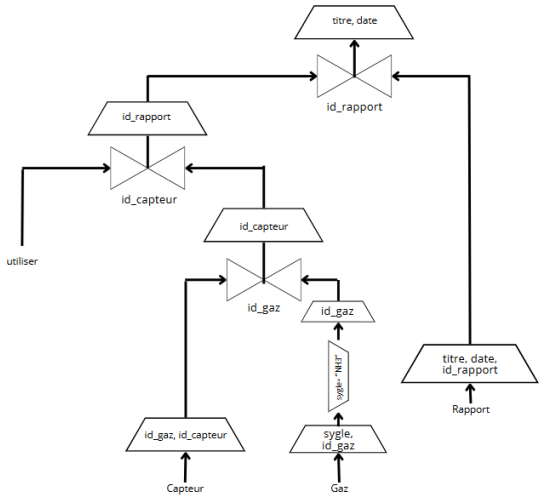


Figure 22 : Arbre Algébrique Requête 7

```
SELECT DISTINCT rapports.Titre_rapport, rapports.Date_publication, gaz.Sigle_gaz
FROM rapports
JOIN utliser  1<->1..n: ON rapports.ID_RAPPORT = utliser.ID_RAPPORT
JOIN capteurs  1..n<->1: ON utliser.ID_CAPTEUR = capteurs.ID_CAPTEUR
JOIN gaz  1..n<->1: ON capteurs.ID_GAZ = gaz.ID_GAZ
WHERE gaz.Sigle_gaz = 'NH3'
ORDER BY rapports.Date_publication DESC;
```

Figure 23 : Code Requête 7

	Titre_rapport	Date_publication	Sigle_gaz
1	Gestion des risques climatiques en entreprise	2024-12-01	NH3
2	Progrès vers la neutralité carbone	2024-09-01	NH3
3	Adaptation climatique dans les secteurs vulnérable	2024-07-01	NH3
4	Transition écologique et transformation des société	2024-05-01	NH3
5	Développement des filières bas carbone	2024-03-01	NH3
6	Stratégies de compensation carbone	2024-02-01	NH3
7	Chaînes logistiques bas carbone	2023-10-01	NH3
8	Impacts du changement climatique sur les océans	2023-08-01	NH3

Figure 24 : Résultat Requête 7

8) *Afficher le mois où la concentration de PFC a été la moins importante pour chaque région :*

Pour cette huitième requête, nous l'avons modifié. Effectivement, nous avons remplacé le PFC avec le méthane de sigle CH4 puisque le PFC n'existe pas dans notre base donnée en raison de l'incohérence du cahier des charges. Dans un premier temps avec la fonction « WITH » nous créons une table fictive contenant la concentration minimale de « CH4 » de chaque région contenue dans une colonne nommé « Minimum_concentration_ppm ». Ensuite, nous sélectionnons les noms des régions, les concentrations minimums de notre table fictive et enfin les dates respectives de ces concentrations. Ces dates sont remaniées afin qu'il n'y soit que le mois qui apparaisse comme demandé et la colonne est renommée en « Mois ». Enfin, dans la condition « WHERE » nous recherchons encore le sigle « CH4 » et les mesures en fonction des résultats de notre table fictive.

```
WITH Concentration AS (
    SELECT régions.Nom_région,
           MIN(mesures.Concentration_ppm) AS Minimum_concentration_ppm
    FROM mesures
    JOIN relever 1<->1..n: ON mesures.ID_MESURE = relever.ID_MESURE
    JOIN capteurs 1..n<->1: ON relever.ID_CAPTEUR = capteurs.ID_CAPTEUR
    JOIN gaz 1..n<->1: ON capteurs.ID_GAZ = gaz.ID_GAZ
    JOIN régions 1..n<->1: ON capteurs.ID_RÉGION = régions.ID_RÉGION
    WHERE gaz.Sigle_gaz = 'CH4'
    GROUP BY régions.Nom_région
)
SELECT régions.Nom_région,
       DATE_FORMAT(mesures.Date_mesure, '%m') as Mois,
       Concentration.Minimum_concentration_ppm
FROM Concentration
JOIN régions ON régions.Nom_région = Concentration.Nom_région
JOIN capteurs 1<->1..n: ON régions.ID_RÉGION = capteurs.ID_RÉGION
JOIN relever 1<->1..n: ON capteurs.ID_CAPTEUR = relever.ID_CAPTEUR
JOIN mesures 1..n<->1: ON relever.ID_MESURE = mesures.ID_MESURE
JOIN gaz 1..n<->1: ON capteurs.ID_GAZ = gaz.ID_GAZ
WHERE gaz.Sigle_gaz = 'CH4'
AND mesures.Concentration_ppm = Concentration.Minimum_concentration_ppm
ORDER BY régions.Nom_région;
```

Figure 25: Code Requête 8

	Nom_région	Mois	Minimum_co…
1	Auvergne-Rhône-Alp…	02	1.01
2	Bourgogne-Franche-…	04	2.73
3	Bretagne	02	2.08
4	Centre-Val de Loire	11	14.35
5	Corse	07	1.6
6	Grand Est	05	1.61
7	Hauts-de-France	11	1.86
8	Île-de-France	10	1.68
9	Normandie	10	4.98
10	Nouvelle-Aquitaine	11	3.03
11	Occitanie	08	1.76
12	Pays de la Loire	06	3.81
13	Provence-Alpes-Côt…	10	5.48

Figure 26 : Résultat Requête 8

9) *Moyenne des concentrations (en ppm) dans la région « Ile-de-France » en 2020, pour chaque gaz étudié :*

Pour la neuvième requête, il faut joindre plusieurs tables, puis utiliser la fonction « AVG » afin de faire la moyenne de concentration de chaque gaz en ppm. À la suite de ça, il faut utiliser la fonction « BETWEEN » pour sélectionner seulement les dates de l'année 2020, puis choisir la région « Île-de-France ». A la toute fin, on ajoute la fonction « GROUP BY » pour ne pas rappeler deux fois le même gaz et pouvoir donc faire la moyenne pour chaque gaz.

```
SELECT Nom_gaz, Sigle_gaz, AVG(concentration_ppm) AS Moyenne_concentration

FROM mesures
JOIN relever 1<->1..n: ON mesures.id_mesure = relever.id_mesure
JOIN capteurs 1..n<->1: ON relever.id_capteur = capteurs.id_capteur
JOIN gaz 1..n<->1: ON capteurs.id_gaz = gaz.id_gaz
JOIN régions 1..n<->1: ON capteurs.id_région = régions.id_région

WHERE mesures.date_mesure BETWEEN '2020-01-01' and '2020-12-01'
AND régions.Nom_région = 'Île-de-France'

GROUP BY gaz.Nom_gaz, Sigle_gaz;
```

Figure 27: Code Requête 9

	Nom_gaz	Sigle_gaz	Moyenne_concentration
1	Protoxyde d'azote	N20	674.7167679220438
2	Ozone troposphérique	O3	773.4709412803253
3	Méthane	CH4	762.6896887024244
4	Hydrofluorocarbures	HFC	708.1270842254162
5	Dioxyde de carbone	C02	714.0385396281878
6	Ammoniac	NH3	760.534579962492

Figure 28: Résultat Requête 9

10) Taux de productivité des agents administratifs de l'agence de Toulouse (le taux est calculé en nombre de rapports écrits par mois en moyenne, sur la durée de leur contrat) :

Pour cette 10e requête, il faut tout d'abord connaître ce que l'on souhaite obtenir avec « SELECT », donc ici on veut l'ID des agents administratifs et taux de productivité de ces agents. Pour ce faire, on doit utiliser « COUNT » pour avoir le nombre de rapports par personnel et diviser ce nombre par sa durée de contrat, qui elle se fait en calculant la différence entre le rapport le plus récent et la date de prise en poste de l'employé. A la suite de cela, il nous faut joindre les tables Personnels, Agences, Écrire et Rapports, puis les sélectionner pour avoir seulement les agents administratifs comme personnel et l'agence de Toulouse. Enfin utiliser « GROUP BY » pour avoir une seule fois l'ID d'un même agent.

<u>Définition fonction :</u>

- TIMESTAMPDIFF(MONTH, date_début, date_fin) : Calcule la durée en mois entre la prise de poste et le dernier rapport.

- NULLIF(..., 0) : Protège contre une division par zéro si la durée est inférieure à un mois.

```sql
SELECT COUNT(Titre_rapport)/NULLIF(TIMESTAMPDIFF(MONTH,personnels.date_prise_poste, CURRENT_DATE()),0) AS Taux_productivité,
    personnels.Nom_personnel,
    personnels.Prénon_personnel,
    agences.Nom_agence

FROM personnels
JOIN agences ON personnels.ID_AGENCE = agences.ID_AGENCE
LEFT JOIN écrire 1<->0..n: ON personnels.ID_AGENT = écrire.ID_AGENT
LEFT JOIN rapports 1..n<->0..1: ON rapports.ID_RAPPORT = écrire.ID_RAPPORT
WHERE personnels.Type_poste = 'agent administratif'
    AND agences.Nom_agence = 'Agence_Toulouse'
GROUP BY
    Nom_personnel, Prénon_personnel, date_prise_poste, Nom_agence;
```

Figure 29: Code Requête 10

	Taux_productivité	Nom_personnel	Prénon_personnel	Nom_agence
1	0.0343	Jean	Martinez	Agence_Toulouse
2	0.4286	Claude	Maillet	Agence_Toulouse
3	0.0210	Kaelig	Grondin	Agence_Toulouse

Figure 30: Résultat Requête 10

11) Pour un gaz donné, liste des rapports contenant des données qui le concernent (on doit pouvoir donner le nom du gaz en paramètre) :

La requête 11 suit le même déroulement que les requêtes précédentes avec une section « FROM », « JOIN » et « WHERE » une petite variance a lieu dans la partie « WHERE » nous permettant de pouvoir déterminer les rapports sur le gaz souhaité : « gaz.Sigle_gaz = :Sigle_gaz ». Cette formule nous permet, avec une interface qui apparait, d'écrire la formule du gaz souhaité.

```
SELECT DISTINCT Sigle_gaz, Titre_rapport
FROM gaz, rapports
JOIN utliser  1<->1..n: on rapports.ID_RAPPORT = utliser.ID_RAPPORT
JOIN capteurs  1..n<->1: on utliser.ID_CAPTEUR = capteurs.ID_CAPTEUR
WHERE gaz.Sigle_gaz = :Sigle_gaz
ORDER BY Titre_rapport;
```

Figure 31: Code Requête 11

Figure 32: Interface Requête 11

	Sigle_gaz	Titre_rapport
1	N20	Accélération de la transition énergétique locale
2	N20	Adaptation climatique dans les secteurs vulnérable
3	N20	Agriculture et adaptation au changement climatique
4	N20	Analyse des politiques climatiques en France
5	N20	Analyse des trajectoires d'émissions territoriales
6	N20	Analyse des vulnérabilités climatiques locales
7	N20	Approches systémiques de la transition écologique

Figure 33: Résultat Requête 11

12) Liste des régions dans lesquelles il y a plus de capteurs que de personnel d'agence :

Chaque donnée se trouvent dans une table différente, il nous faut faire la jonction entre la table régions, agences, personnel et capteurs. Ensuite, il faut comparer le nombre de capteur dans une région avec le nombre de personnel en utilisant la fonction « COUNT » (pour le nombre) et « > » comme indice de comparaison. Ensuite on affiche les régions qui valide cette dernière information.

```
SELECT nom_région,
COUNT(DISTINCT capteurs.ID_CAPTEUR) AS Nombre_capteurs,
COUNT(DISTINCT personnels.ID_AGENT) AS Nombre_personnels
FROM régions
JOIN capteurs 1<->1..n: ON régions.ID_RÉGION = capteurs.ID_RÉGION
JOIN agences 1<->1..n: ON régions.ID_RÉGION = agences.ID_RÉGION
JOIN personnels ON personnels.ID_AGENCE = agences.ID_AGENCE
GROUP BY régions.ID_RÉGION, Nom_région
HAVING COUNT(DISTINCT capteurs.ID_CAPTEUR) > COUNT(DISTINCT personnels.ID_AGENT);
```

Figure 34: Code Requête 12

	nom_région	Nombre_capteurs	Nombre_personnels
1	Corse	12	9
2	Nouvelle-Aquitaine	72	18
3	Bourgogne-Franche-Comté	48	15
4	Hauts-de-France	30	15
5	Auvergne-Rhône-Alpes	72	12
6	Provence-Alpes-Côte d'Azur	36	9
7	Pays de la Loire	30	5
8	Centre-Val de Loire	36	6
9	Île-de-France	48	18
10	Bretagne	24	12
11	Normandie	30	6
12	Grand Est	60	8
13	Occitanie	78	7

Figure 35: Résultat Requête 12

Peuplement de la base de données

1) **Origine des Données :**

Pour remplir notre base de données, nous avons utilisé une intelligence artificielle génératrice de données, à savoir ChatGPT. Cette IA (Intelligence Artificielle) nous a permis de produire rapidement des données fictives ainsi que cohérentes, adaptées à la structure et aux besoins établis pour notre base de données.

Nous avons rédigé des prompts précis mais concis pour chaque table de notre base de données. Ils étaient tous conçu pour générer un ensemble de données respectant les normes, le contexte et les valeurs souhaités (Exemple : Les dates de publication des rapports ou bien les types de poste).

Cela nous a permis d'obtenir un jeu de données assez conséquents tout en assurant une certaine cohérence logique par rapport au cahier des charges prédéfini.

2) **Gestion des Clés Primaires et Secondaires :**

Les clés primaires et secondaires ont été créées manuellement afin de garantir le bon fonctionnement des liens entre les différentes tables de notre base de données. De plus, cela nous a aussi permis de choisir parfois certaines liaisons dans le but de varier les données en sortie de nos requêtes.

3) **Les principales difficultés rencontrer sont les suivantes :**

- Les données générées par l'IA étaient en volume limitées, ce qui veut dire que nous avions dû envoyer des requêtes sur plusieurs jours afin de compléter toutes nos tables.
- Nous avons perdu du temps sur la génération de nos données car sur certaines tables nous devions réaliser des modifications pour cause d'une mauvaise compréhension du cahier des charges de notre entreprise.
- Lors peuplement de notre fichier Excel, les dates ou encore les nombres n'étaient pas aux bons formats (Exemple : JJ/MM/AAAA ≠ AAAA-MM-JJ ou bien 123,45 ≠ 123.45) ce qui nous a compliqué la tâche lors du peuplement de la base de données.
- Lors de l'implantation des données dans notre base, il nous fallait d'abord remplir toutes les tables sans les clés secondaires et ensuite revenir sur chaque table pour mettre les clés secondaires.
- Enfin, nous avions dû gérer manuellement la plupart des clés étrangères afin d'éviter les erreurs de correspondances et nous permettant de choisir les liens des données.

Malgré ses différentes perturbations techniques et logistiques, l'utilisation de l'IA nous a permis de gagner un temps précieux pour le peuplement de la base de données.

Exploitation de la base de données

L'une des dernières tâches à réalisé était la création d'au moins deux comptes utilisateurs afin de simuler l'accès à notre de base de données. C'est pourquoi, nous avons réalisé un compte « Utilisateur » et un autre « Administrateur ». Pour ce faire, il nous a fallut exécuter ces deux commandes respectives :

1) Création du compte administrateur :

Le compte administrateur a tous les droits sur la base de données. Il peut modifier les données des tables, la structure ainsi que les utilisateurs de la base.

```
CREATE USER 'Admin' IDENTIFIED BY 'MDPADMIN';
GRANT ALL PRIVILEGES ON `livrable bdd`.* TO 'admin'@'localhost';
FLUSH PRIVILEGES;
```

Figure 36:Requête de création de compte Administrateur

2) Création du compte utilisateur :

Le compte utilisateur n'a que certains droits sur la base de données. Il peut simplement lire les données des tables ainsi qu'effectuer seulement des requêtes de recherche de données.

```
CREATE USER 'User' IDENTIFIED BY 'MDPUSER';
GRANT SELECT ON `livrable bdd`.* TO 'user'@'localhost';
FLUSH PRIVILEGES;
```

Figure 37: Requête de création de compte Utilisateur

Conclusion

Dans ce livrable, nous avons donc réalisé les 12 requêtes sous forme de code SQL, les 7 premières requêtes avaient été demandés précédemment sous forme d'arbre algébrique, cela nous a donc permit de réaliser plus facilement les premiers codes. La base de données a été générée et peuplée à partir de plusieurs étapes : la réalisation d'un Excel avec les différentes tables permettant d'importer les données et la création d'un code afin de faire apparaitre chacune des différentes requêtes.

La création de cette base de données « test » fonctionnelle va donc nous permettre de lancer la réalisation de l'outil final permettant de stocker et interroger des données sur la qualité de l'air dans les grandes villes de France.

Annexe