Une image contenant capture d’écran, Graphique, cercle, conception

Le contenu généré par l’IA peut être incorrect.Une image contenant capture d’écran, Graphique, cercle, conception

Le contenu généré par l’IA peut être incorrect.

Projets donnés et modélisation :

**Livrable 2**

**Equipe :**

Belgacem Omaima ; Droussi Wael ; Duvivier Mathilde ; Leo Bogreau ; Cousin Benjamin

**Dictionnaire de données :**

Après lecture et étude du besoin nous avons retranscris les données dans le dictionnaire en déduisant également certaine donnée évidente.



Code : Nom des attributs de la base de données

Désignation : Définition et rôle de l’attribut

Nature : Chaque donnée est élémentaire car elle est unique et indivisible

Type : Le type de la donnée (entier, texte, date, décimal, etc…)

Taille : La taille maximum alloué à la donnée

Min-Max : Le minimum et le maximum de valeurs de clé possible

On regroupe les attributs ensemble pour former des entités (famille de donné) et notre but est de lié ces entités entre elles pour faire des associations, c’est un modèle conceptuel de donnée.



**Modèle conceptuel de donnée :**

Après association des attributs nous obtenons les entités (en jaune) et nous les lions entre elles avec des associations (en bleu).

Une image contenant texte, capture d’écran, diagramme, ligne

Le contenu généré par l’IA peut être incorrect.

Chaque entité comporte des attributs qui lui sont propre et qui ne dépende d’aucune autre entité, et chaque attribut est unique et qu’on ne peut pas décomposer, on dit que l’attribut est atomique.

Associations avec cardinalités :

Agents 🡨🡪 Agences : Chaque agents est employés pour 1 agence.  
 Chaque agences a 1 ou n employés qui travaille.

Agents 🡨🡪 Rapports : Chaque agents rédige 1 ou n rapports.

Chaque rapport est rédigé par 1 ou n agents.

Agents 🡨🡪 Capteurs : Chaque agents relevé 1 ou n capteurs.

Chaque capteur est relevé par 1 agent.

Agences 🡨🡪 Capteurs : Chaque agence gère 1 ou n capteurs.

Chaque capteur est géré par 1 agence.

Rapports 🡨🡪 Mesure : Chaque rapport contient 0 ou n mesure.

Chaque mesure est dans 0 ou n rapport

Capteurs 🡨🡪 Mesure : Chaque capteur produit 1 ou n mesure.

Chaque mesure est produite par 1 capteur.

Pour que le modèle conceptuel soit correct et reconnu comme telle il doit respecter des règles de normalisation qui vise à pouvoir assurer la nature même de la base de données et que le modèle conceptuel de donnée puisse être traduit en modèle logique de donnée puis en modèle physique.

Normalisation :

1ère forme normale (1FN) : Ce niveau de normalisation garantit que chaque colonne de vos données ne contient que des valeurs atomiques. Dans ce contexte, les valeurs atomiques signifient que chaque entrée d'une colonne est indivisible. C'est comme si l'on disait que chaque cellule d'une feuille de calcul ne devait contenir qu'une seule information. La 1NF garantit l'atomicité des données, chaque cellule de colonne ne contenant qu'une seule valeur et chaque colonne ayant un nom unique.

2ème forme normale (2FN) : Élimine les dépendances partielles en veillant à ce que les attributs autres que la clé ne dépendent que de la clé primaire. Cela signifie essentiellement qu'il doit y avoir une relation directe entre chaque colonne et la clé primaire, et non entre les autres colonnes

3ème forme normale (3FN) : Supprime les dépendances transitives en veillant à ce que les attributs non-clés ne dépendent que de la clé primaire. Ce niveau de normalisation s'appuie sur la norme 2NF.

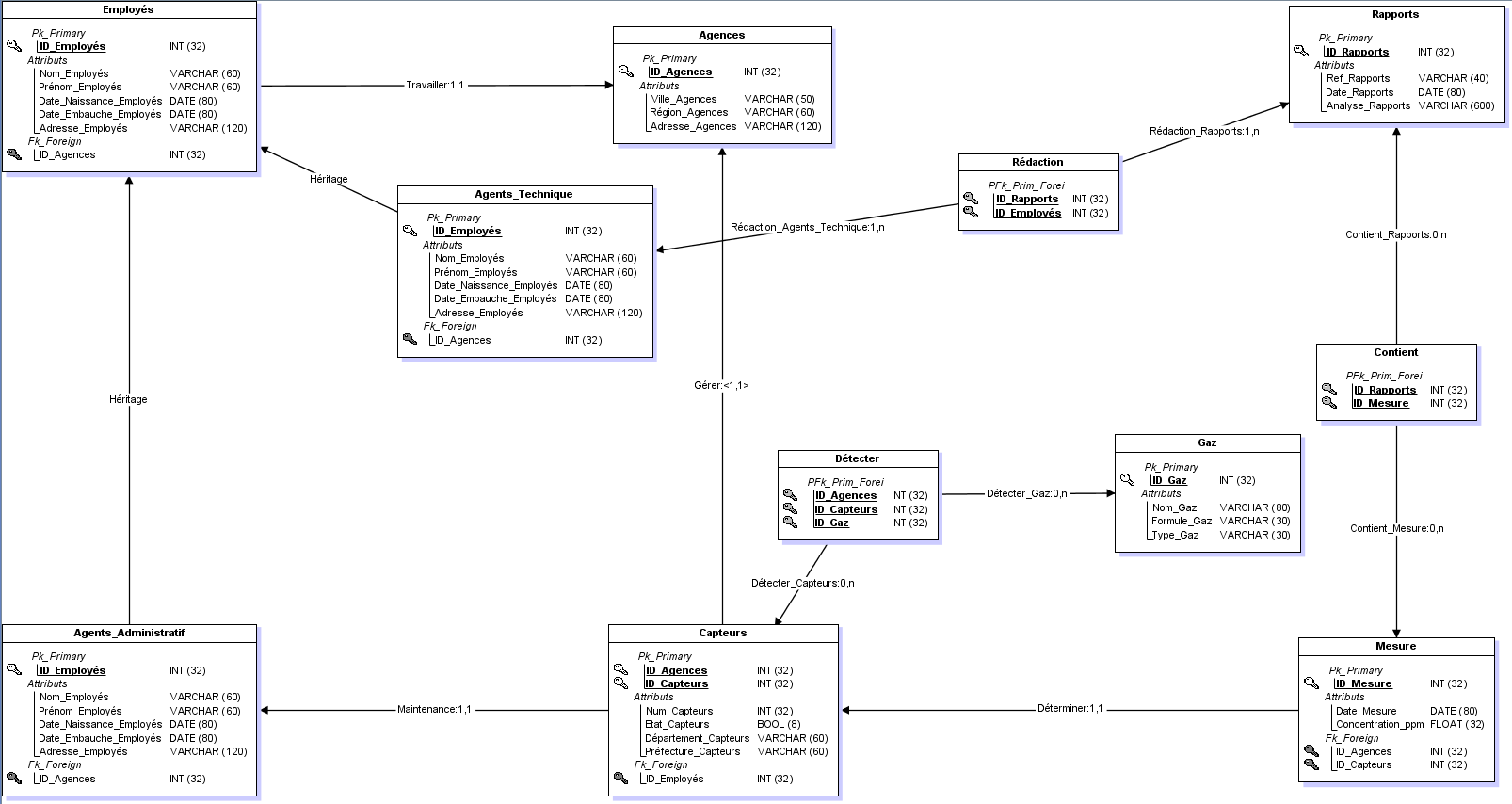
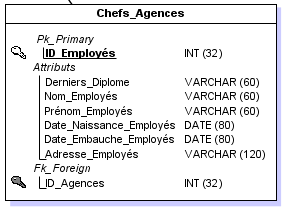
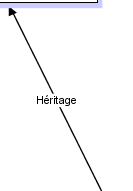
Etant donné que les trois premières règles de normalisation sont respectées nous pouvons désormais passer au modèle logique de donnée. Pour réaliser celui-ci il y à quelques règles et changement à respecter :

|  |  |  |
| --- | --- | --- |
| **Élément MCD** | **Élément MLD** | **Explication** |
| Entité | Relation | Chaque entité devient une relation (ou table logique). |
| Propriété d'entité | Attribut | Chaque propriété devient un attribut de la relation. |
| Identifiant d'entité | Clé primaire (PK) | L’identifiant devient la clé primaire de la relation. |
| Association (1,1) | Clé étrangère dans l'entité de l'autre côté | L'identifiant de l'entité côté (1,1) devient clé étrangère dans l'autre. |
| Association (0, N) – (1,1) | Clé étrangère côté N | L’identifiant de l'entité côté (1,1) est inséré comme clé étrangère dans l'entité côté (0, N). |
| Association (0, N) – (0, N) | Nouvelle relation associative | Une nouvelle relation est créée avec les identifiants des deux entités. |
|  |  | Ces identifiants deviennent la **clé primaire composée** et des **clés étrangères**. |
| Association avec attributs | Nouvelle relation associative avec attributs | Les attributs de l’association sont intégrés à la nouvelle relation. |
| Spécialisation / généralisation | Plusieurs possibilités selon le modèle choisi | (Héritage par inclusion, table par entité fille, etc.) |

Il nous reste alors seulement à appliquer ces changements et réaliser le modèle logique de données



**Modèle logique de donnée :**



Ainsi nous obtenons le modèle logique de données qui est le modèle se rapprochant le plus du modèle physique.

Explication en détail :

* Agences : PK = ID\_Agences et ne possède pas de clé étrangère car ne dépend d’aucune autre table.
* Employés : PK = ID\_Employés et FK = ID\_Agences car les employés travaillent pour l’agence (association).
* Chefs\_Agences : PK = ID\_Employés et FK = ID\_Agences car il hérite des attributs de la table employés.
* Agents\_Techniques : PK = ID\_Employés et FK = ID\_Agences car il hérite des attributs de la table employés.
* Agents\_Administratif : PK = ID\_Employés et FK = ID\_Agences car il hérite des attributs de la table employés.
* Capteurs : PK = ID\_Agences, ID\_Capteurs et FK = ID\_Employés car chaque capteur dépend de l’agence à laquelle il est assigné et que chaque capteur est entretenu par un agent administratif.
* Détecter : PK = ID\_Agences, ID\_Capteurs, ID\_Gaz car c’est la table regroupant les capteurs et les informations concernant le gaz détecter par celui-ci.
* Gaz : PK = ID\_Capteurs et ne possède pas de clé étrangère car ne dépend pas d’une autre table.
* Mesure : PK = ID\_Mesure et FK = ID\_Agences, ID\_Capteurs car chaque mesure dépend du capteur et donc de l’agence.
* Contient : PK = ID\_Rapports, ID\_Mesure car il contient les informations sur les rapports et les mesures rassemblés.
* Rédaction : PK = ID\_Rapports, ID\_Employés car il contient les informations sur le rapport ainsi que le ou les agents techniques qui l’ont rédigé.
* Rapports : PK = ID\_Rapports et ne contient pas de clé étrangère car ne dépend d’aucune autre table.

Mais avant de réaliser le modèle physique des données il faut prendre en compte quelques règles de conversion :

|  |  |  |
| --- | --- | --- |
| **Élément MLD** | **Élément MPD** | **Explication** |
| Relation | Table | Chaque relation devient une table physique dans la base de données. |
| Attribut | Colonne | Chaque attribut devient une colonne de la table. |
| Clé primaire | PRIMARY KEY | L’attribut (ou groupe d’attributs) identifiant la relation devient la clé primaire. |
| Clé étrangère | FOREIGN KEY | Les identifiants d’autres relations utilisées deviennent des clés étrangères. |
| Relation avec clé composée | PRIMARY KEY composée | Une relation associative possède souvent une clé primaire composée de deux clés étrangères. |
| Types de données | Typage SQL (INT, VARCHAR, DATE, etc.) | Chaque attribut doit être associé à un type physique selon le SGDB utilisé. |
| Contraintes | NOT NULL, UNIQUE, CHECK, etc. | Les règles métiers sont traduites en contraintes d’intégrité. |
| Nom des relations et attributs | Respect de la syntaxe du SGBD | Par exemple, éviter les espaces, caractères spéciaux, mots réservés. |

Il nous reste alors seulement à appliquer ces règles et réaliser le modèle physique de données

**Modèle physique de donnée :**



Désormais que le modèle physique il nous reste à réaliser quelques requêtes pour s’assurer que l’algèbre relationnelle s’applique bien à la base de données et que nous pouvons l’exploiter pleinement.

**Jointure :**

Avant de réaliser les jointures et les arbres algébrique de requête nous devons d’abord comprendre et assimiler chaque opérateur de l’algèbre relationnelle :

Explication en détail :

Sélection =

La sélection génère une relation regroupant exclusivement toutes les occurrences de la relation R qui satisfont l’expression logique prédicat.

Projection =

La projection consiste à supprimer les attributs autres que A1, …An d’une relation et à éliminer les n-uplets en double apparaissant dans la nouvelle relation.

Produit cartésien =

Le produit cartésien est une opération portant sur deux relations R1 et R2 et qui construit une troisième relation regroupant exclusivement toutes les possibilités de combinaison des occurrences des relation R1 et R2.

Jointure =

La jointure est une opération portant sur deux relations R1 et R2 qui construit une troisième relation regroupant exclusivement toutes les possibilités de combinaison des occurrences des relations R1 et R2 qui satisfont l’expression logique E.

L’union =

L’union est une opération portant sur deux relations R1 et R2 ayant le même schéma et construisant une troisième relation constituée de n-uplet appartenant à chacune des deux relations R1 et R2.

L’intersection =

L’intersection est une opération portant sur deux relations R1 et R2 ayant le même schéma et construisant une troisième relation dont les n-uplets sont constitués de ceux appartenant aux deux relations.

L’agrégation =

L'opération est une opération fondamentale qui permet de calculer des résumés statistiques sur des ensembles de données en produisant une nouvelle relation en sortie.



**1ère requête** : La liste de l’ensemble des Agences

Ici on affichera seulement la ville de chaque agence au lieu de l’identifiant par soucis de compréhension.

π Ville\_Agences

|

δ (élimination des doublons)

|

Agences

SELECT DISTINCT Ville\_Agences

FROM Agences;

**2ème requête** : Liste de l'ensemble du personnel technique de l'agence de Bordeaux

Ici on affichera seulement le nom et le prénom du personnel technique de l’agence de Bordeaux.

π e.ID\_Employes, e.Nom\_Employes, e.Prenom\_Employes, e.Date\_Naissance\_Employes,

e.Date\_Embauche\_Employes, e.Adresse\_Employes

|

⋈ e.ID\_Agences = a.ID\_Agences

/ \

⋈ e.ID\_Employes = at.ID\_Employes σ a.Ville\_Agences = 'Bordeaux'

/ \ |

Employes e Agences a

\

Agents\_Technique at

SELECT e.ID\_Employes, e.Nom\_Employes, e.Prenom\_Employes, e.Date\_Naissance\_Employes,

e.Date\_Embauche\_Employes, e.Adresse\_Employes

FROM Employes e

JOIN Agents\_Technique at ON e.ID\_Employes = at.ID\_Employes

JOIN Agences a ON e.ID\_Agences = a.ID\_Agences

WHERE a.Ville\_Agences = 'Bordeaux';

**3ème requête** : Nombre total de capteurs déployé

Pour cela on part du principe que les capteurs de chaque région sont numérotés de 1 à n sans trous ni doublons.

SELECT COUNT(ID\_Capteurs) FROM Capteurs;

π Total\_Capteurs

|

γ COUNT(\*)→Total\_Capteurs

|

Capteurs

**4ème requête** : Liste des rapports publiés entre 2018 et 2022

On affiche le titre des rapports publiées entre 2018 et 2022 (compris).

SELECT Titre\_Rapports, Date\_Rapports FROM Rapports

WHERE YEAR(Date\_Rapports) BETWEEN 2018 AND 2022

ORDER BY Date\_Rapports;

τ Date\_Rapports (tri)

|

π Titre\_Rapports, Date\_Rapports (projection)

|

σ YEAR(Date\_Rapports) BETWEEN 2018 AND 2022 (sélection)

|

Rapports

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

On affiche les concentrations en mai et juin 2023 dans les 3 régions indiqué.

SELECT

a.Region\_Agences AS "Région",

a.Ville\_Agences AS "Ville Agence",

c.Departement\_Capteurs AS "Département",

m.Date\_Mesure,

m.Concentration\_ppm AS "CH4 (ppm)",

g.Nom\_Gaz

FROM

Mesure m

JOIN Capteurs c ON m.ID\_Capteurs = c.ID\_Capteurs

JOIN Agences a ON c.ID\_Agences = a.ID\_Agences

JOIN Detecter d ON c.ID\_Capteurs = d.ID\_Capteurs

JOIN Gaz g ON d.ID\_Gaz = g.ID\_Gaz

WHERE

g.Formule\_Gaz = 'CH₄'

AND a.Region\_Agences IN ('Île-de-France', 'Bretagne', 'Occitanie')

AND m.Date\_Mesure BETWEEN '2023-05-01' AND '2023-06-30'

ORDER BY

a.Region\_Agences,

m.Date\_Mesure;

π Région, "Ville Agence", "Département", Date\_Mesure, "CH4 (ppm)", Nom\_Gaz

|

τ a.Region\_Agences, m.Date\_Mesure (tri)

|

σ g.Formule\_Gaz = 'CH₄'

∧ a.Region\_Agences IN ('Île-de-France', 'Bretagne', 'Occitanie')

∧ m.Date\_Mesure BETWEEN '2023-05-01' AND '2023-06-30' (sélection)

|

⋈ (jointure complète)

/ | | \

⋈ ⋈ ⋈ ⋈

/ \ / \ / \ / \

Mesure m Capteurs c Agences a Detecter d Gaz g

(m.ID\_Capteurs = c.ID\_Capteurs) (c.ID\_Agences = a.ID\_Agences)

(c.ID\_Capteurs = d.ID\_Capteurs) (d.ID\_Gaz = g.ID\_Gaz)

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

Requête pour afficher exclusivement le nom des agents techniques dans les régions spécifiées et pour la durée demandée.

SELECT DISTINCT e.Nom\_Employes, e.Prenom\_Employes

FROM Employes e

JOIN Agents\_Technique at ON e.ID\_Employes = at.ID\_Employes

JOIN Capteurs c ON at.ID\_Employes = c.ID\_Employes

JOIN Detecter d ON c.ID\_Capteurs = d.ID\_Capteurs

JOIN Gaz g ON d.ID\_Gaz = g.ID\_Gaz

WHERE g.Type\_Gaz = 'GESI'

ORDER BY e.Nom\_Employes, e.Prenom\_Employes;

π e.Nom\_Employes, e.Prenom\_Employes (projection finale)

|

δ (élimination des doublons DISTINCT)

|

τ e.Nom\_Employes, e.Prenom\_Employes (tri ORDER BY)

|

σ g.Type\_Gaz = 'GESI' (sélection WHERE)

|

⋈ (jointure complète)

/ | | \

⋈ ⋈ ⋈ ⋈

/ \ / \ / \ / \

Employes e Agents\_Technique at Capteurs c Detecter d Gaz g

(e.ID\_Employes = at.ID\_Employes) (at.ID\_Employes = c.ID\_Employes)

(c.ID\_Capteurs = d.ID\_Capteurs) (d.ID\_Gaz = g.ID\_Gaz)

**7ème requête** : Titres et dates des rapports concernant des concentrations de NH3, classés par ordre antichronologique.

Requête qui liste les titres et dates des rapports traitant des concentrations d'ammoniac (NH₃), triés du plus récent au plus ancien :

SELECT r.Titre\_Rapports, r.Date\_Rapports

FROM Rapports r

JOIN Contient c ON r.ID\_Rapports = c.ID\_Rapports

JOIN Mesure m ON c.ID\_Mesure = m.ID\_Mesure

JOIN Detecter d ON m.ID\_Capteurs = d.ID\_Capteurs

JOIN Gaz g ON d.ID\_Gaz = g.ID\_Gaz

WHERE g.Nom\_Gaz = 'Ammoniac' OR g.Formule\_Gaz = 'NH₃'

ORDER BY r.Date\_Rapports DESC;

π r.Titre\_Rapports, r.Date\_Rapports (projection finale)

|

τ r.Date\_Rapports DESC (tri)

|

σ g.Nom\_Gaz = 'Ammoniac' ∨ g.Formule\_Gaz = 'NH₃' (sélection)

|

⋈ (jointure complète)

/ | | \

⋈ ⋈ ⋈ ⋈

/ \ / \ / \ / \

Rapports r Contient c Mesure m Detecter d Gaz g

(r.ID\_Rapports = c.ID\_Rapports) (c.ID\_Mesure = m.ID\_Mesure)

(m.ID\_Capteurs = d.ID\_Capteurs) (d.ID\_Gaz = g.ID\_Gaz)

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

Pour cela on cherche les mois avec les concentrations les plus faible en Perfluorocarbures

et avec les jointures nécaissaire on affiche les nom des région correspondante

ainsi que le mois de la mesure et la valeur de celle-ci.

SELECT

region\_data.Région,

region\_data.Mois,

region\_data.Concentration\_minimale

FROM (

SELECT

a.Region\_Agences AS Région,

MONTH(m.Date\_Mesure) AS Mois,

MIN(m.Concentration\_ppm) AS Concentration\_minimale,

ROW\_NUMBER() OVER (PARTITION BY a.Region\_Agences ORDER BY MIN(m.Concentration\_ppm) ASC) AS rang

FROM

Agences a

JOIN

Capteurs c ON a.ID\_Agences = c.ID\_Agences

JOIN

Mesure m ON c.ID\_Capteurs = m.ID\_Capteurs

JOIN

Detecter d ON c.ID\_Capteurs = d.ID\_Capteurs

JOIN

Gaz g ON d.ID\_Gaz = g.ID\_Gaz

WHERE

g.Nom\_Gaz = 'Perfluorocarbures' -- ou g.ID\_Gaz = 6

GROUP BY

a.Region\_Agences, MONTH(m.Date\_Mesure)

) AS region\_data

WHERE

region\_data.rang = 1

ORDER BY

region\_data.Concentration\_minimale ASC;

**9ème requête** :  Moyenne des concentrations (en ppm) dans la région « Ile-de-France » en 2020,

pour chaque gaz étudié.

Cette requête calcule la concentration moyenne annuelle (en ppm) de chaque gaz mesuré en Île-de-France,

en regroupant les résultats par gaz et par année, puis les trie chronologiquement.

SELECT

g.Nom\_Gaz AS 'Gaz étudié',

YEAR(m.Date\_Mesure) AS Année,

ROUND(AVG(m.Concentration\_ppm), 2) AS 'Moyenne (ppm)'

FROM

Gaz g

JOIN

Detecter d ON g.ID\_Gaz = d.ID\_Gaz

JOIN

Capteurs c ON d.ID\_Capteurs = c.ID\_Capteurs

JOIN

Agences a ON c.ID\_Agences = a.ID\_Agences

JOIN

Mesure m ON c.ID\_Capteurs = m.ID\_Capteurs

WHERE

a.Region\_Agences LIKE '%le-de-France%'

AND m.Date\_Mesure IS NOT NULL

AND m.Concentration\_ppm IS NOT NULL

GROUP BY

g.Nom\_Gaz, YEAR(m.Date\_Mesure)

HAVING

AVG(m.Concentration\_ppm) IS NOT NULL

ORDER BY

Année, g.Nom\_Gaz;

**10ème requête** : 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)

Cette requête analyse la productivité des employés administratifs de Toulouse en calculant leur taux de rapports rédigés par mois de service, trié du plus au moins productif

SELECT

e.ID\_Employes,

e.Nom\_Employes,

e.Prenom\_Employes,

e.Date\_Embauche\_Employes,

-- Calcul du nombre de mois depuis l'embauche (en inclure les mois partiels)

TIMESTAMPDIFF(MONTH, e.Date\_Embauche\_Employes, IFNULL(MAX(r.Date\_Rapports), CURDATE())) AS Mois\_Service,

-- Nombre total de rapports rédigés

COUNT(DISTINCT red.ID\_Rapports) AS Nombre\_Rapports,

-- Taux de productivité (rapports/mois) arrondi à 2 décimales

ROUND(COUNT(DISTINCT red.ID\_Rapports) /

GREATEST(TIMESTAMPDIFF(MONTH, e.Date\_Embauche\_Employes, IFNULL(MAX(r.Date\_Rapports), CURDATE())), 1), 2) AS Taux\_Productivite

FROM

Employes e

JOIN

Agents\_Administratif aa ON e.ID\_Employes = aa.ID\_Employes

JOIN

Agences a ON e.ID\_Agences = a.ID\_Agences

LEFT JOIN

Redaction red ON e.ID\_Employes = red.ID\_Employes

LEFT JOIN

Rapports r ON red.ID\_Rapports = r.ID\_Rapports

WHERE

a.Ville\_Agences = 'Toulouse'

GROUP BY

e.ID\_Employes, e.Nom\_Employes, e.Prenom\_Employes, e.Date\_Embauche\_Employes

HAVING

Mois\_Service > 0 -- Exclure les employés embauchés ce mois-ci

ORDER BY

Taux\_Productivite DESC;

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

Cette requête analyse les rapports d'étude sur l'Ammoniac (paramétrable), en fournissant pour chaque rapport : sa localisation, le nombre de mesures associées, et les statistiques de concentration (min, max, moyenne).

SET @nom\_gaz = 'Ammoniac'; -- Paramètre à modifier

SELECT

r.ID\_Rapports,

r.Titre\_Rapports,

r.Ref\_Rapports,

r.Date\_Rapports,

a.Region\_Agences AS Region,

g.Nom\_Gaz AS Gaz\_Etudie,

COUNT(DISTINCT m.ID\_Mesure) AS Nombre\_Mesures,

MIN(m.Concentration\_ppm) AS Concentration\_Min,

MAX(m.Concentration\_ppm) AS Concentration\_Max,

ROUND(AVG(m.Concentration\_ppm), 2) AS Concentration\_Moyenne

FROM

Rapports r

JOIN

Contient c ON r.ID\_Rapports = c.ID\_Rapports

JOIN

Mesure m ON c.ID\_Mesure = m.ID\_Mesure

JOIN

Capteurs cap ON m.ID\_Capteurs = cap.ID\_Capteurs

JOIN

Detecter d ON cap.ID\_Capteurs = d.ID\_Capteurs

JOIN

Gaz g ON d.ID\_Gaz = g.ID\_Gaz

JOIN

Agences a ON r.ID\_Agences = a.ID\_Agences

WHERE

g.Nom\_Gaz = @nom\_gaz

GROUP BY

r.ID\_Rapports, r.Titre\_Rapports, r.Ref\_Rapports, r.Date\_Rapports, a.Region\_Agences, g.Nom\_Gaz

ORDER BY

r.Date\_Rapports DESC;

**12ème requête** : Liste des régions dans lesquelles il y a plus de capteurs que de personnel d’agence.

Cette requête compare le nombre de capteurs et d'employés par région, en identifiant les déséquilibres

(plus de capteurs que de personnel) et les classe par écart décroissant.

SELECT

a.Region\_Agences AS Region,

COUNT(DISTINCT c.ID\_Capteurs) AS Nombre\_Capteurs,

COUNT(DISTINCT e.ID\_Employes) AS Nombre\_Personnel,

CASE

WHEN COUNT(DISTINCT c.ID\_Capteurs) > COUNT(DISTINCT e.ID\_Employes) THEN 'Plus de capteurs'

ELSE 'Plus de personnel'

END AS Comparaison

FROM

Agences a

LEFT JOIN

Capteurs c ON a.ID\_Agences = c.ID\_Agences

LEFT JOIN

Employes e ON a.ID\_Agences = e.ID\_Agences

GROUP BY

a.Region\_Agences

HAVING

COUNT(DISTINCT c.ID\_Capteurs) > COUNT(DISTINCT e.ID\_Employes)

ORDER BY

(COUNT(DISTINCT c.ID\_Capteurs) - COUNT(DISTINCT e.ID\_Employes)) DESC;

**Peuplement de la base de données :**

Pour peupler la base de données nous avons choisis des valeurs arbitraire et cohérente

pour chacune des tables. Nous avons parfois fait appel a de nouvelles intelligences artificielle

pour pouvoir générer un grand nombre de données rapidement (l’IA Mockaroo), mais cela n’enlèves rien

au fait que toutes les clés étrangère ont du être associé une par une pour garder d\*plusieurs cohérence

géographique et technique.

Employés:

Pour peupler la table employés nous avons utiliser le site Mockaroo qui nous à permis de générer 100 noms, prénoms, date de naissance, date d’embauche et adresse différente qui par soucis de facilité n’ont pas de cohérence direct avec la localisation de l’agence dans laquelle travaille l’employés. Et nous avons rentrés manuellement la clé étrangère id\_agence pour chacun des employés (choisis arbitrairement car aucune importance).

Agences:

Pour peupler la table agences nous avons utilisé les 18 régions de France (métropolitaine et DOM-TOM), ainsi que les chefs lieux de chacune de ces régions et enfin une adresse cohérente au chef lieux pour l’agence.

Chefs d’agences:

Ici nous avons seulement choisis 18 employés de chaque agence différente parmis la table employés et nous avons fait correspondre leurs diplomes (choisis arbitrairement) avec la clé étrangère id\_employés et ainsi chaque agences avait un chef.

Agents administratif:

Nous avons choisis 39 employés en les répartissant de manière la plus équitable dans chaque agences et leurs avons attribués un id\_employés pour qu’ils appartiennent à la table agents administratif.

Agents Techniques:

Nous avons pris les 43 employés restant et leurs avons attribué le role d’agents techniques en faisant bien correspondre leurs id\_employés.

Capteurs:

Pour peupler la table capteurs nous avons fait appel à une intelligence artificielle (ChatGPT) pour faire correspondre chaque département avec la préfecture dans laquelle il se trouve ainsi que la régions de l’agence de base. Nous aurions pu le faire manuellement hors cela aurait été trop long et fastidieux pour obtenir 100 capteurs. Mais en revanche nous avons du quand même rentrés manuellement chaque ID\_Employés des agents techniques qui se trouvait dans la même région que l’agence qui entretient le capteurs.

Rapports :

Nous avons choisis arbitrairement chaque valeurs sans forcément de logique adaptés car ce n’était pas nécaissaire à la cohérence des données. Hormis les titres qui devaient tout de même avoir de la cohérence avec le sujet ainsi que l’analyse qui devait ensuite être cohérente avec le titre.

Mesures :

Nous avons encore une fois utilisé l’intelligence artificielle en lui demandant de nous fournir des dates aléatoires (plutôt récente) ainsi que des valeurs décimale qui pourrait représenté des données de capteurs cohérentes et nous avons ensuite répartis chaque mesure a un capteur choisis arbitrairement.

Gaz :

Nous avons 7 gaz à effet de serre naturelle et 12 gaz a effect de serre industrielle, nous avons bien évidemment pris de véritable gaz ainsi que leurs véritable formule nous avons ensuite fait correspondre tout ca dans différentes table contient et détecter.

**Conclusion :**

En conclusion nous avons construit entièrement construit une base de données en utilisant la méthode Merise.  
En effect on a commencé par la lecture et l’analyse du besoin pour déterminer le dictionnaire de données.

Ensuite à l’aide du dictionnaire de données nous avons pu déduire des relations entre chaque entité et leurs fournir des attributs et ainsi formé le modèle conceptuel de donnée.

Après cela nous avons déterminer le modèle logique de donnée en se basant sur le modèle conceptuel de donnée et les règles de normalisation s’appliquant à cette conversion.

Puis on a pu déterminer le modèle physique de donnée grâce au langage SQL.

Pour enfin peupler notre base de données pour pouvoir la manipuler à notre guise.

Sur la base de cette base de données nous avons pu effectuer plusieurs requêtes ciblées afin d’obtenir des informations et des données spécifiques à l’aide de jointure et de l’algèbre relationnel pour créer à chaque fois des arbres algébriques de requête.