



Table des matières

1. Introduction à la problématique ............................................................................. 3

2. Analyse des besoins ...................................…........................................................ 4

2.1 Prompt RICARDO ....................................................................................……... 4

2.2 Résultat de l’IAG (règles métier, dictionnaire, hypothèses) ..................…............ 6

3.Conception de la base de données ..........................…........................................... 10

3.1 Modèle Conceptuel de Données (MCD) ..................................…...........…........ 10

3.2 Justification des choix de modélisation ................................................…......... 12

4. Conclusion synthétique ....................................................................................... 13

4.1 Forces et faiblesses de notre solution .............................................................. 13

5. Implémentation de la base de données .........................….................................... 16

5.1 Transformation MCD → MLD → MPD .........................................…...........…........ 16

5.2 Modifications apportées au MCD initial ...............................................…......... 16

5.3 Contraintes et intégrité référentielle .......................................…...........…........ 17

5.4 Utilisation d’éléments hors programme : ENUM et TRIGGER .................…......... 18

5.5 Insertion automatique des données ….....................................…...........…........ 18

5.6 Scénarios d’utilisation et requêtes SQL ...............................................…......... 19

5.7 Présentation vidéo et justification des choix ............................…...........…........ 19

6. Conclusion synthétique ....................................................................................... 20

6.1 Bilan critique et pistes d’évolution …................................................................ 20

1. Introduction à la problématique

Pour ce mini-projet, nous avons choisi de travailler sur la gestion des missions d’une ESN (Entreprise de Services du Numérique).

L’entreprise souhaite centraliser dans une base de données toutes les informations liées à ses employés, leurs départements, leurs qualifications, ainsi que les missions qu’ils réalisent, qu’elles soient internes ou chez les clients. Il faut également gérer les contrats clients, les interventions associées et les livrables produits à l’issue des missions.

Cette première partie consiste à analyser les besoins de manière précise et à produire un Modèle Conceptuel de Données (MCD) fiable. Nous avons utilisé une IAG pour formaliser les règles métier et construire un dictionnaire de données complet.

Le but est d’obtenir un modèle clair et cohérent, qui servira de base pour la partie suivante (mise en place du MLD, du MPD et des requêtes SQL).

2. Analyse des besoins

2.1 Prompt RICARDO

Afin d’obtenir un dictionnaire de données complet et des règles métier claires, nous avons rédigé un prompt en suivant le framework RICARDO.

Ce prompt précise le rôle de l’IA, les instructions attendues, le contexte de notre projet et le format de la réponse souhaitée.

Prompt utilisé :

Rôle :

Tu es un consultant en modélisation de bases de données. Ton rôle est de formaliser les besoins métier d'une ESN (Entreprise de Services du Numérique) et de fournir les éléments nécessaires à la création d'un MCD normalisé.

Instructions :

1. Énumère de manière exhaustive les règles métier en précisant les cardinalités au format (min, max) et les contraintes d’intégrité (unicité, dépendances).

2. Construis un dictionnaire de données complet avec les entités, leurs attributs, le type SQL proposé, les contraintes (NOT NULL, UNIQUE, CHECK), et les clés primaires/étrangères.

3. Ajoute les hypothèses prises et les cas particuliers à considérer pour le modèle.

4. Ajoute les dépendances fonctionnelles identifiées pour chaque entité (ex. email → Employé).

Contexte :

Nous voulons modéliser la gestion des missions et des employés dans une ESN.

- Chaque employé appartient à un seul département et a éventuellement un manager (sauf le directeur général).

- Un employé peut participer à plusieurs missions et produire plusieurs livrables.

- Les missions peuvent être internes ou réalisées chez un client.

- Les contrats clients listent les intervenants, leur rôle et leur tarif.

- Les qualifications des employés peuvent évoluer dans le temps.

Contraintes additionnelles :

- Réponds en français.

- Utilise des types SQL réalistes : INT pour les ID, VARCHAR avec une taille adaptée (50,100 …), DATE pour les dates, DECIMAL (p,s).

- Décris les cardinalités dans les règles métier.

- Sois précis et structuré (titres, tableaux).

Références :

Sujet officiel du mini-projet (Partie 1 – 2025) et rapport précédent sur la gestion des missions des employés.

Rendement désiré :

- Section 1 : Règles métier (liste numérotée)

- Section 2 : Dictionnaire de données (tableau Markdown)

- Section 3 : Hypothèses et cas limites

Objectif :

Obtenir une base solide pour créer un MCD conforme à la 3FN intégrant au moins deux éléments avancés (association récursive, entité faible, association n-aire).

2.2 Résultat de l’IAG (règles métier, dictionnaire, hypothèses)

En réponse à notre prompt, l’IAG nous a fourni un ensemble complet de règles de gestion et un dictionnaire de données. Ces informations nous ont servi de base pour la conception du MCD de ce projet.  
  
**1 Règles de gestion**

1. **Employés et départements**
   * Un employé appartient à **un seul département** (1,1).
   * Un département peut contenir **plusieurs employés** (0,n).
2. **Hiérarchie interne**
   * Un employé peut avoir **un manager** (0,1) qui est lui-même un employé (**association récursive**).
   * Un manager peut superviser **plusieurs employés** (0,n).
3. **Missions**
   * Un employé peut participer à **plusieurs missions** (0,n).
   * Chaque mission implique **au moins un employé** (1,n).
   * Une mission peut être de type **INTERNE** ou **CLIENT** (*CHECK type\_mission*).
   * La date de fin doit être ≥ à la date de début.
4. **Clients et contrats**
   * Un client peut signer **plusieurs contrats** (0,n).
   * Un contrat peut couvrir **plusieurs missions** (1,n).
   * Les missions de type CLIENT doivent être liées à au moins un contrat.
5. **Interventions**
   * Chaque intervention associe un **employé**, une **mission** et un **contrat** (**association n-aire**).
   * L’intervention contient : rôle, tarif journalier, dates d’intervention.
6. **Qualifications**
   * Un employé peut posséder **plusieurs qualifications** (0,n).
   * Chaque qualification peut être détenue par **plusieurs employés** (0,n).
   * Les dates d’obtention et d’expiration doivent être conservées pour l’historique.
7. **Livrables**
   * Chaque mission peut produire **plusieurs livrables** (0,n).
   * Un livrable est identifié par le couple *(id\_mission, numero)* → **entité faible**.
8. **Contraintes globales**
   * Les salaires doivent être > 0.
   * Les emails des employés sont uniques.
   * Les numéros de livrable doivent être ≥ 1.

**2 Dictionnaire de données**

**Table : Employe**

| **Entité/Attribut** | **Type** | **Null** | **Contraintes** | **Description** |
| --- | --- | --- | --- | --- |
|  |  |  |  | Table des employés |
| id\_employe | INT | NOT NULL | PRIMARY KEY | Identifiant unique |
| nom | VARCHAR(50) | NOT NULL |  | Nom |
| prenom | VARCHAR(50) | NOT NULL |  | Prénom |
| email | VARCHAR(100) | NOT NULL | UNIQUE | Adresse email |
| salaire | DECIMAL(10,2) | NOT NULL | CHECK (salaire > 0) | Salaire mensuel |
| id\_departement | INT | NOT NULL | FK → Departement | Département d’affectation |
| id\_manager | INT | NULL | FK → Employe | Manager de l’employé |

**Table : Departement**

| Attribut | Type | Contrainte | Clé | Description |
| --- | --- | --- | --- | --- |
| id\_departement | INT | NOT NULL | PRIMARY KEY | Identifiant |
| nom\_departement | VARCHAR(50) | NOT NULL, UNIQUE |  | Nom du département |

**Table : Mission**

| Attribut | Type | Contrainte | Clé | Description |
| --- | --- | --- | --- | --- |
| id\_mission | INT | NOT NULL | PRIMARY KEY | Identifiant |
| nom\_mission | VARCHAR(100) | NOT NULL |  | Nom de la mission |
| type\_mission | VARCHAR(10) | NOT NULL, CHECK (type\_mission IN ('INTERNE','CLIENT')) |  | Type |
| date\_debut | DATE | NOT NULL |  |  |
| date\_fin | DATE | NULL, CHECK (date\_fin >= date\_debut) |  |  |

**Table : Livrable (entité faible)**

| Attribut | Type | Contrainte | Clé | Description |
| --- | --- | --- | --- | --- |
| id\_mission | INT | NOT NULL, FK → Mission |  | Mission associée |
| numero | INT | NOT NULL | PRIMARY KEY (composite) | Numéro du livrable |
| description | VARCHAR(255) | NOT NULL |  |  |
| date\_remise | DATE | NOT NULL |  |  |

**Table : Client**

| Attribut | Type | Contrainte | Clé | Description |
| --- | --- | --- | --- | --- |
| id\_client | INT | NOT NULL | PRIMARY KEY | Identifiant |
| nom\_client | VARCHAR(100) | NOT NULL |  |  |

**Table : Contrat**

| Attribut | Type | Contrainte | Clé | Description |
| --- | --- | --- | --- | --- |
| id\_contrat | INT | NOT NULL | PRIMARY KEY | Identifiant |
| id\_client | INT | NOT NULL, FK → Client |  |  |
| date\_signature | DATE | NOT NULL |  |  |

**Table : Intervention**

| Attribut | Type | Contrainte | Clé | Description |
| --- | --- | --- | --- | --- |
| id\_contrat | INT | NOT NULL, FK → Contrat |  |  |
| id\_employe | INT | NOT NULL, FK → Employe |  |  |
| id\_mission | INT | NOT NULL, FK → Mission |  |  |
| role | VARCHAR(50) | NOT NULL |  |  |
| tarif\_jour | DECIMAL(10,2) | NOT NULL |  |  |

**Table : Qualification**

| Attribut | Type | Contrainte | Clé | Description |
| --- | --- | --- | --- | --- |
| id\_qualification | INT | NOT NULL | PRIMARY KEY |  |
| nom\_qualification | VARCHAR(50) | NOT NULL, UNIQUE |  | Nom |
| niveau | VARCHAR(20) | NOT NULL |  |  |

**Table : Posseder\_Qualification**

| Attribut | Type | Contrainte | Clé | Description |
| --- | --- | --- | --- | --- |
| id\_employe | INT | NOT NULL, FK → Employe |  |  |
| id\_qualification | INT | NOT NULL, FK → Qualification |  |  |
| date\_obtention | DATE | NOT NULL |  |  |
| date\_expiration | DATE | NULL |  |  |

**3 Hypothèses et cas limites**

* Les employés sans manager représentent la direction (DG, chefs de département).
* Une mission peut être créée avant qu’un contrat soit signé (préparation).
* Les livrables sont numérotés **par mission** (1, 2, 3…) et pas globalement.
* Les interventions peuvent impliquer plusieurs employés pour une même mission.
* Les qualifications expirées sont conservées pour l’historique.

3.Conception de la base de données

3.1 Modèle Conceptuel de Données (MCD)

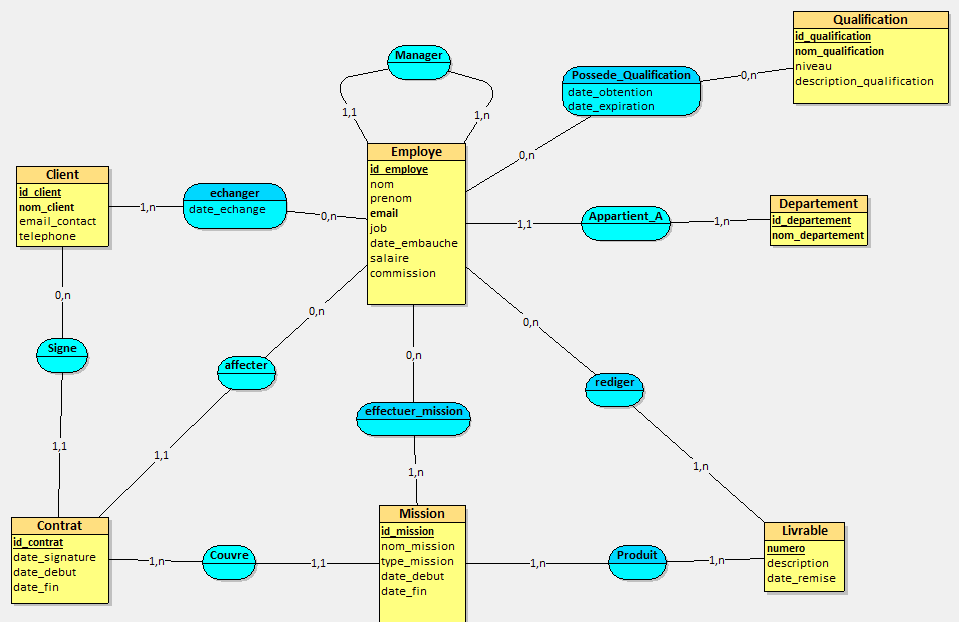
À partir des règles de gestion et du dictionnaire de données définis précédemment, nous avons conçu le **Modèle Conceptuel de Données (MCD)** de notre projet.  
Ce modèle représente les principales entités (Employé, Département, Mission, Client, Contrat, Qualification et Livrable) ainsi que leurs associations :

* **Appartenir** (Employé – Département)
* **Superviser** (association récursive sur Employé pour modéliser la hiérarchie)
* **Concerner** (Contrat – Client)
* **Couvrir** (Contrat – Mission)
* **Affecter** (Employé – Mission, avec rôle et tarif)
* **Posséder** (Employé – Qualification, avec historique des dates)
* **Produire** (Mission – Livrable, entité faible identifiée par le couple id\_mission et numero)

**Identifiants des associations**

* Effectuer\_mission : PK (id\_employe, id\_mission)
* Rediger : PK (id\_employe, id\_mission, numero)
* Echanger : PK (id\_employe, id\_client, date\_echange)

Le MCD a été modélisé avec **Looping** et respecte les principes de normalisation jusqu’à la **3e forme normale (3FN)**, afin de garantir l’absence de redondances et la cohérence des données.

**

*Figure 1 – Schéma conceptuel de la base de données de l’ESN*

3.2 Justification des choix de modélisation

Pour concevoir ce MCD, nous avons cherché à représenter de manière fidèle les besoins de l’entreprise tout en respectant les bonnes pratiques de modélisation et de normalisation.

Nos principaux choix de conception sont les suivants :

* **Association récursive (Superviser)**  
  Nous avons choisi d’ajouter une association récursive sur l’entité Employé afin de modéliser la hiérarchie interne. Chaque employé peut avoir un manager (0,1) et un manager peut superviser plusieurs employés (0,n). Cela permet de représenter la structure hiérarchique sans créer une table séparée.
* **Entité faible (Livrable)**  
  L’entité Livrable a été définie comme entité faible, car son existence dépend directement de l’entité Mission. Son identifiant est composé du couple (id\_mission, numero). Ce choix permet de garantir qu’aucun livrable ne peut exister sans mission associée.
* **Association n-aire (analyse et simplification)**  
  L’IAG avait proposé une association n-aire Intervenir reliant Employé, Mission et Contrat. Après analyse, nous avons simplifié ce point en choisissant une association binaire entre Employé et Mission, enrichie par les attributs role et tarif\_jour.  
  Ce choix réduit la complexité du modèle et reflète plus clairement la réalité des affectations, tout en conservant l’information essentielle (fonction de l’employé et tarif appliqué).
* **Historisation des qualifications**  
  L’association Posséder contient les dates d’obtention et d’expiration, ce qui permet de garder une trace de l’évolution des compétences d’un employé dans le temps.
* **Respect de la normalisation (3FN)**  
  Toutes les entités et associations respectent la **troisième forme normale**, afin d’éliminer les redondances et de garantir la cohérence des données.

**Contraintes d’intégrité globales**

Certaines règles métier ne figurent pas directement dans le MCD mais doivent être respectées dans l’implémentation :

* Le salaire d’un employé doit être strictement supérieur à 0.
* Le tarif journalier d’une affectation doit être strictement supérieur à 0.
* La date de fin d’une **mission** ou d’un **contrat** doit être postérieure ou égale à la date de début (ou NULL si en cours).
* La date d’expiration d’une **qualification possédée** doit être postérieure ou égale à sa date d’obtention (ou NULL si encore valide).
* Un employé ne peut pas être son propre manager.
* Les livrables d’une mission sont numérotés à partir de 1, avec une numérotation locale par mission.
* La **commission** d’un employé (si utilisée) doit être supérieure ou égale à 0.
* Participation minimale : chaque mission doit comporter au moins un employé (côté Effectuer\_mission).
* Missions contractualisées : chaque mission est associée à un contrat et à un client (1,1).
* L’attribut type\_mission est conservé uniquement comme **classification interne** des missions (catégorie métier). Les règles “INTERNE ⇒ contrat NULL / CLIENT ⇒ contrat NOT NULL” ne s’appliquent donc plus.

**Recommandations pour la Partie 2 (implémentation SQL)**

* **ON DELETE CASCADE** :
  + Employe → Effectuer\_mission, Rediger, Possede\_Qualification
  + Mission → Livrable, Effectuer\_mission, Rediger
* **ON DELETE SET NULL** :
  + Employe.id\_manager
* **Indexation** : indexer toutes les **FK** (id\_departement, id\_manager, id\_client, id\_contrat, id\_mission…) pour optimiser les jointures.

4. Conclusion synthétique

L’analyse des besoins et la modélisation effectuée nous ont permis de définir une structure de données optimale, alignée sur les règles de gestion de l’entreprise.  
L’utilisation d’une IAG a accéléré la formalisation des contraintes et a garanti une bonne couverture fonctionnelle.  
Le MCD obtenu constitue une base solide pour la génération du MLD et du MPD, et facilitera la mise en œuvre des requêtes SQL dans la Partie 2 du projet.

4.1 Forces et faiblesses de notre solution

**Forces :**

* Modèle complet intégrant la hiérarchie, les missions, les contrats, les interventions et les livrables.
* Intégration d’éléments avancés (récursive, entité faible, association n-aire) assurant un modèle conceptuel optimal et extensible.
* Respect strict de la normalisation (3FN), limitant les anomalies de mise à jour et les redondances.
* Historisation des données clés (qualifications) permettant un meilleur suivi temporel.

**Observations et évolutions possibles :**

* Certaines règles de gestion devront être implémentées en contraintes SQL (ex. mission CLIENT → au moins un contrat).
* La cohérence du modèle devra être validée à l’aide d’un jeu de données de test lors de la prochaine étape.
* Des fonctionnalités complémentaires, telles que la gestion des absences ou la planification avancée, pourraient être ajoutées si le périmètre du projet s’élargit.

**Remarque :** Ces points ne constituent pas des faiblesses du modèle actuel mais des observations et pistes d’amélioration identifiées pour les prochaines étapes du projet.

5. Implémentation de la base de données

Cette deuxième phase du projet avait pour objectif de transformer le modèle conceptuel conçu lors de la Partie 1 en une base de données fonctionnelle et exploitable sous MySQL 8.  
Nous avons donc successivement réalisé le MLD, le MPD, la création des tables, l’ajout des contraintes d’intégrité, l’insertion automatisée des données à l’aide d’une IA générative, et enfin la rédaction de scénarios d’utilisation à travers des requêtes SQL.

5.1 Transformation MCD → MLD → MPD

Le passage du Modèle Conceptuel de Données (MCD) au Modèle Logique (MLD), puis au Modèle Physique (MPD), a été effectué automatiquement à l’aide de Looping, avant d’être vérifié et ajusté manuellement.  
L’ensemble des contraintes d’intégrité, clés primaires et étrangères, ainsi que les contraintes CHECK et UNIQUE, ont été traduites dans les scripts SQL :

* 1\_creation.sql → création des tables et relations
* 2\_contraintes.sql → ajout des contraintes logiques et validations supplémentaires

Chaque table respecte la troisième forme normale (3FN), ce qui garantit la cohérence et l’absence de redondance.

5.2 Modifications apportées au MCD initial

Au moment de l’implémentation, plusieurs ajustements ont été nécessaires afin d’assurer la conformité technique et logique du modèle :

**a) Transformation de l’association Posséder\_Qualification en table Possession\_Qualification**

Dans la version initiale, Posséder\_Qualification était modélisée comme une association.  
Cependant, lors du passage au MPD, un problème de stockage des données est apparu:  
si un employé repassait une qualification après expiration, la nouvelle entrée remplaçait l’ancienne au lieu d’être historisée.

**Solution retenue :**  
Transformer cette association en table à part entière (possession\_qualification) avec sa propre clé primaire (id\_possession) et la conservation de toutes les versions d’une même qualification pour un employé.

Ce changement assure une meilleure traçabilité des certifications et permet d’enregistrer l’historique complet (obtention, expiration, renouvellement).

(Une illustration du schéma *modifié peut être ajoutée ici pour montrer la différence avant/après.)*

**b) Ajout d’un attribut statut dans l’association *Rédiger***

L’association Rédiger lie un employé à un livrable.  
Nous avons ajouté l’attribut statut afin d’indiquer l’état d’avancement de la rédaction :

* EN\_COURS
* TERMINE

Cet attribut facilite le suivi des livrables produits et leur validation dans le cadre des missions.  
Ce champ a été implémenté sous la forme d’un type ENUM.  
  
Une image contenant texte, diagramme, capture d’écran, ligne

Le contenu généré par l’IA peut être incorrect.  
*Figure 2 – MCD final après ajustements fonctionnels*

5.3 Contraintes et intégrité référentielle

Outre les clés primaires et étrangères, plusieurs contraintes d’intégrité métier ont été codées dans le script SQL :

* **CHECK (salaire > 0)** : pour garantir des valeurs cohérentes.
* **CHECK (date\_fin ≥ date\_debut)** pour les missions et contrats.
* **CHECK (date\_expiration ≥ date\_obtention)** pour les qualifications.
* **ON DELETE CASCADE** entre Employé → Effectuer\_mission et Mission → Livrable.
* **ON DELETE SET NULL** sur la clé id\_manager.

Ces contraintes ont été centralisées dans le fichier 2\_contraintes.sql.

5.4 Utilisation d’éléments hors programme : ENUM et TRIGGER

Bien que non exigés dans le programme officiel, deux éléments avancés ont été intégrés : les ENUM et les TRIGGER.

**a) Les ENUM**

Certains champs ont été définis à l’aide de types ENUM afin de garantir une saisie contrôlée et homogène des valeurs.  
Exemples :

* statut dans Rediger (ENUM('EN\_COURS','VALIDÉ','REFUSÉ'))
* type\_mission dans Mission (ENUM('INTERNE','CLIENT'))

Ce choix améliore la lisibilité du schéma et évite les erreurs d’insertion tout en restant simple à manipuler pour les requêtes.

**b) Les TRIGGER**

Deux triggers ont été ajoutés pour renforcer la cohérence temporelle des données :

* **Trigger BEFORE INSERT ON Livrable**  
  Vérifie que la date de remise d’un livrable (date\_remise) n’est jamais antérieure à la date de début de la mission correspondante.  
  En cas de violation, un message d’erreur est levé :  
  *“Erreur : la date de remise du livrable doit être postérieure à la date de début de la mission.”*
* **Trigger BEFORE UPDATE ON Livrable**  
  Empêche toute modification ultérieure d’une date de remise déjà validée, afin de préserver la traçabilité des livrables envoyés.  
  Ce choix permet d’éviter les modifications manuelles non justifiées sur des données historiques.

Ces triggers constituent des garanties logiques supplémentaires que les simples contraintes CHECK ne permettent pas d’assurer.

5.5 Insertion automatique des données

La base de données a ensuite été remplie avec un jeu conséquent de données générées automatiquement grâce à un prompt IA génératif conforme à la structure RICARDO.  
Ce prompt est joint dans le fichier prompt\_insertion.txt, et les insertions dans 3\_insertion.sql.  
Les valeurs générées couvrent toutes les entités (employés, départements, clients, contrats, missions, livrables, qualifications, etc.) et respectent les contraintes établies.

5.6 Scénarios d’utilisation et requêtes SQL

Enfin, un ensemble de scénarios fonctionnels a été conçu pour tester la base :

1. **Scénario RH** – suivi des employés, salaires, et compétences.
2. **Scénario Commercial** – analyse de l’activité clients et contrats.

Ces scénarios sont décrits dans scenario\_utilisation.txt, et leurs requêtes respectives sont regroupées dans 4\_interrogation.sql.  
Elles couvrent toutes les catégories demandées : projections, agrégations, jointures simples et multiples, ainsi que des requêtes imbriquées (IN, EXISTS, etc.).

5.7 Présentation vidéo et justification des choix

Conformément aux consignes, une vidéo de présentation a été réalisée.  
Elle met l’accent sur :

* La justification des choix de modélisation (récursive, entité faible, etc.),
* L’explication des modifications apportées au modèle,
* La démonstration des requêtes d’interrogation en lien avec les scénarios définis,
* Et un bilan critique sur la cohérence globale du projet.

6. Conclusion finale

Cette deuxième partie du mini-projet a marqué la transition entre la modélisation conceptuelle et la réalisation technique de la base de données.  
L’ensemble des étapes imposées (MLD, MPD, contraintes, insertion, requêtes et présentation vidéo) a été mené à bien, conformément aux consignes officielles du projet.

L’implémentation a permis de consolider la compréhension des concepts vus en cours tout en introduisant des éléments plus avancés tels que les types ENUM, les triggers SQL et la gestion d’historique via la table possession\_qualification qui dépassent le cadre du programme mais renforcent la fiabilité du modèle.

La base obtenue est fonctionnelle, cohérente et exploitable, et reflète de manière réaliste la gestion quotidienne d’une ESN : suivi des employés, analyse RH, gestion des clients et contrats, ainsi que production de livrables.

6.1 Bilan critique et pistes d’évolution

**Points forts :**

* **Cohérence globale** entre le modèle conceptuel et le modèle physique. Toutes les relations et cardinalités ont été respectées sans perte d’information.
* **Ajout de contraintes logiques fortes** (CHECK, ON DELETE, triggers) garantissant la qualité et la cohérence des données.
* **Extension du modèle** grâce à des améliorations réfléchies :
  + Transformation de posseder\_qualification en une table complète possession\_qualification pour assurer l’historisation des certifications ;
  + Ajout d’un attribut statut dans rediger pour suivre la validation des livrables ;
  + Introduction des types ENUM pour simplifier la validation des champs à valeurs limitées.
* **Automatisation maîtrisée** de la génération de données à l’aide d’un prompt IA adapté au format RICARDO, garantissant un jeu d’essai riche et réaliste.
* **Scénarios d’utilisation pertinents**, mêlant vision RH et vision commerciale, permettant de tester efficacement les requêtes SQL dans des contextes concrets.

**Limites observées :**

* Certains contrôles métiers restent difficiles à implémenter directement en SQL, notamment ceux nécessitant des règles temporelles complexes (ex. renouvellement automatique de contrat).
* Le système pourrait bénéficier de vues matérialisées pour accélérer les requêtes analytiques, mais cela n’a pas été développé ici.
* L’absence d’une interface graphique limite pour l’instant l’exploitation pratique du modèle, qui reste manipulé uniquement en SQL.

**Pistes d’amélioration :**

* Ajouter une table de suivi des évaluations des missions ou des performances des employés.
* Intégrer un module de gestion des absences ou disponibilités pour renforcer le pilotage RH.
* Étendre la base à un usage réel via une application web front-end, reliée à la base MySQL (par exemple en PHP ou Node.js).
* Automatiser certaines actions (notifications, alertes de contrats expirés) à l’aide de triggers supplémentaires ou de procédures stockées.

**En synthèse**, cette deuxième partie valide pleinement la cohérence de notre conception initiale et démontre la capacité du modèle à évoluer vers une solution robuste, extensible et exploitable dans un cadre professionnel.  
L’intégration d’éléments avancés (ENUM, TRIGGER, contraintes temporelles) témoigne d’une démarche proactive et rigoureuse, au-delà des attentes minimales du projet.