Rapport INFO0505

Cabinet Cosmique

Valentin MAROT

Table des matières

Introduction

Contexte

Dans le cadre de ses activités, le cabinet cosmique (cabinet de conseil en ressources humaines) joue un rôle essentiel dans le recrutement de cadres pour des entreprises clientes. Ces entreprises, en quête de talents spécifiques pour des postes qualifiés, s'adressent au cabinet pour les accompagner dans la recherche, la sélection et l'évaluation de candidats adaptés. Le cabinet, ayant acquis une certaine reconnaissance dans le secteur, attire aussi bien des candidatures spontanées que des partenariats avec des institutions d'enseignement, notamment des écoles et des universités qui forment les futurs cadres. Grâce à ce positionnement, le cabinet dispose d'un vivier de candidats potentiels qui enrichit ses dossiers et permet de répondre efficacement aux besoins de ses clients.

Pour répondre aux attentes des entreprises et améliorer sa gestion interne, le cabinet de conseil a décidé de développer une base de données centralisée. Celle-ci doit non seulement permettre de gérer les informations sur les candidats, les missions, et les entreprises, mais aussi de faciliter les relations avec les instituts partenaires. Le projet de base de données est ainsi conçu pour structurer, automatiser et sécuriser les processus de recrutement, tout en optimisant le suivi et la sélection des candidats. Cela constitue un avantage concurrentiel important pour le cabinet dans un marché en constante évolution où la rapidité et la précision de recrutement sont essentielles.

Problématique

Dans un secteur compétitif comme celui des ressources humaines, le cabinet de conseil doit pouvoir traiter, gérer et actualiser efficacement un volume conséquent de données. Les entreprises s'attendent à ce que le cabinet identifie les meilleurs candidats pour chaque poste en un minimum de temps. De plus, pour faciliter l'embauche des jeunes diplômés, le cabinet collabore avec divers instituts, ajoutant un autre niveau de complexité à la gestion des profils. L'objectif est donc de créer une base de données structurée permettant une recherche efficace des candidats et une gestion fluide des missions de recrutement.

Objectifs du projet

Ce projet vise à doter le cabinet de conseil d'un outil de gestion efficace, centralisant toutes les informations nécessaires pour appuyer les missions de recrutement de cadres. En rassemblant les données des entreprises clientes, les missions de recherche, ainsi que les profils des candidats et des étudiants, nous souhaitons simplifier l'accès et l'organisation de ces informations. L'objectif est de faciliter

l’identification rapide des candidats idéaux pour chaque mission, tout en permettant une mise à jour continue des dossiers au fil des interactions.

Ainsi, nous espérons améliorer le suivi des missions, renforcer les partenariats avec les entreprises et les instituts, et permettre un processus de recrutement plus fluide et ciblé.

Analyse des besoins

Identification des acteurs

Pour comprendre le fonctionnement du cabinet de conseil et de sa base de données, nous devons identifier les différents acteurs impliqués :

Entreprises clientes : Ce sont les entreprises qui confient des missions de recrutement au cabinet. Elles partagent leurs besoins en recrutement et s’attendent à ce que le cabinet leur propose des profils adéquats pour les postes vacants.

Candidats : Ce sont des cadres ou des professionnels qui postulent pour des missions via le cabinet. Ils peuvent avoir soumis leurs dossiers par candidature spontanée, via des annonces, ou avoir été approchés par le cabinet pour des postes spécifiques.

Instituts (universités et écoles) : Ils fournissent au cabinet des listings d’étudiants et des informations sur les diplômes proposés. Les conventions établies avec ces instituts permettent au cabinet de disposer d’un réservoir de futurs candidats potentiels.

Étudiants : Ce sont des profils issus des instituts avec lesquels le cabinet collabore. Bien que non encore actifs dans le marché professionnel, ils constituent des candidats potentiels pour les missions, notamment pour les postes de jeunes diplômés.

Fonctions et missions : Les fonctions représentent les rôles que les candidats peuvent occuper, tandis que les missions sont des postes spécifiques à pourvoir dans les entreprises. Les fonctions et missions définissent les besoins en compétences et qualifications que la base de données doit pouvoir gérer.

Scénarios d’utilisation

Voici quelques exemples d’utilisation pour illustrer comment la base de données répondra aux besoins du cabinet :

Recherche de candidats pour une mission : Lorsqu’une entreprise confie une mission au cabinet, le chargé de recrutement peut utiliser la base de données pour identifier les candidats potentiels en fonction de critères tels que le diplôme, la fonction actuelle, la mobilité géographique, et les compétences.

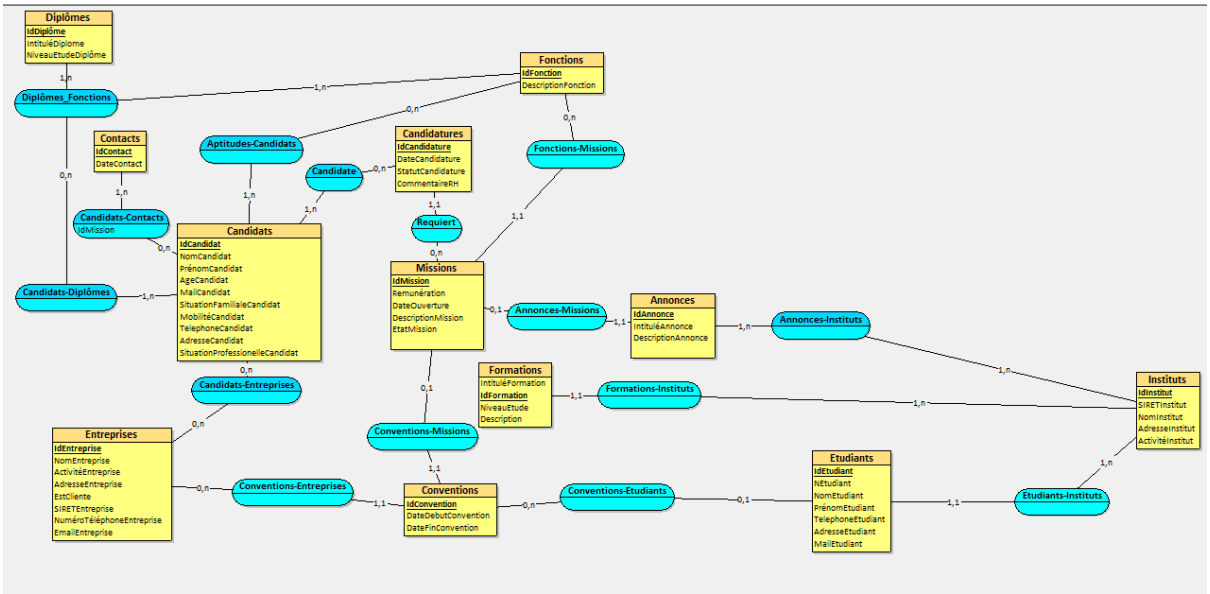
Gestion des candidatures spontanées : Le cabinet reçoit régulièrement des candidatures spontanées. Ces candidats sont intégrés à la base de données et peuvent être facilement retrouvés pour les nouvelles missions correspondant à leurs profils.

Consultation des informations sur les entreprises : La base de données contient les informations des entreprises clientes et non clientes, permettant au cabinet de garder une trace des missions précédentes et des profils d’entreprises avec lesquelles il est en contact, même lorsque certaines missions sont clôturées.

Gestion des étudiants et des conventions : En collaboration avec les instituts, le cabinet obtient les listings d’étudiants inscrits aux différents diplômes. Cela permet au cabinet de recruter directement des jeunes diplômés et de gérer les informations des étudiants, en particulier ceux qui peuvent devenir candidats par la suite.

Modèle conceptuel de données

Présentation du MCD



Explication des entités

1. **Candidats** : Cette entité regroupe les informations personnelles et professionnelles des candidats. Les attributs incluent :

- *IdCandidat* : identifiant unique du candidat.
- *Nom* et *Prénom* : informations d'identité.
- *Age*, *Situation familiale*, et *Mobilité* : précisent la situation personnelle et la disponibilité géographique.
- *Situation professionnelle* : indique le statut actuel (en recherche, salarié, etc.).

2. **Entreprises** : Cette entité contient les données des entreprises clientes et non clientes du cabinet. Les attributs principaux sont :

- *IdEntreprise* : identifiant unique de l'entreprise.
- *Nom*, *Activité*, *AdresseEntreprise* : informations descriptives de l'entreprise.
- *EstCliente* : précise si l'entreprise est cliente du cabinet.

3. **Missions** : Les missions représentent les projets de recrutement confiés au cabinet. Elles incluent :

- *IdMission* : identifiant unique de la mission.
- *Remuneration* : rémunération prévue pour le poste.
- *DateOuverture*, *Description*, *EtatMission* : définissent la date de début, le descriptif de la mission, et son statut (ouverte, fermée).

4. **Fonctions** : Les fonctions correspondent aux postes types que le cabinet de conseil recrute. Les attributs principaux sont :

- *IdFonction* : identifiant unique de la fonction.
- *IntituleFonction* : nom du poste (ex. Ingénieur Logiciel, Chef de Projet).

5. **Diplomes** : Cette entité représente les différents diplômes que peuvent détenir les candidats ou que les étudiants sont en train de préparer. Les attributs sont :

- *IdDiplome* : identifiant unique du diplôme.
- *NomDiplome* et *NiveauEtude* : décrivent le titre du diplôme et le niveau d'étude associé (Bac, Bac+3, Bac+5, etc.).

6. **Formations** : Elle regroupe les cursus académiques proposés par les instituts et suivis par les étudiants. Attributs :

- *IdFormation* : identifiant unique de la formation.
- *NomFormation*, *NiveauFormation* : nom et niveau de la formation.

7. **Instituts** : Les instituts sont les écoles ou universités avec lesquelles le cabinet collabore pour accéder à des listings d'étudiants. Attributs principaux :

- *IdInstitut* : identifiant unique de l'institut.
- *NomInstitut*, *AdresseInstitut*, *ActiviteInstitut* : précisent le nom, l'adresse et le type d'activité.

8. **Etudiants** : Entité représentant les étudiants inscrits dans les instituts partenaires. Les attributs incluent :

- *IdEtudiant* : identifiant unique de l'étudiant.
- *Nom*, *Prenom*, *NumeroEtudiant*, et *Telephone* : informations personnelles et de contact.

9. **Annonces** : Les annonces représentent les offres d'emploi diffusées par le cabinet pour des missions spécifiques. Les attributs sont :

- *IdAnnonce* : identifiant unique de l'annonce.
- *IntituleAnnonce* : description brève de l'offre d'emploi.

10. **Conventions** : Cette entité contient les accords entre le cabinet et les instituts pour recruter les étudiants diplômés. Attributs :

- *IdConvention* : identifiant unique de la convention.
- *DateDebut*, *DateFin* : période de validité de la convention.

11. **Candidatures** : Les candidatures représentent les réponses des candidats aux annonces diffusées par le cabinet. Les attributs incluent :

- *IdCandidature* : identifiant unique de la candidature.
- *DateCandidature*, *StatutCandidature*, *CommentaireCandidature* : précisent la date, le statut actuel (en cours, accepté, refusé) et les notes relatives à la candidature.

Contraintes et cardinalités

Les relations entre ces entités sont définies pour respecter les besoins de gestion des missions et des candidats :

Entreprise-Mission : une entreprise peut confier plusieurs missions au cabinet (relation 1,N), mais chaque mission est liée à une seule entreprise (1,1).

Fonction-Mission : chaque mission correspond à une seule fonction (1,1), mais une fonction peut être recherchée pour plusieurs missions (1,N).

Diplome-Fonction : un diplôme peut correspondre à plusieurs fonctions (1,N) et chaque fonction peut exiger plusieurs diplômes potentiels (N,N).

Candidat-Fonction : un candidat peut se sentir apte pour plusieurs fonctions (1,N), et chaque fonction peut être occupée par plusieurs candidats (N,N).

Institut-Etudiant : chaque étudiant est inscrit dans un seul institut (1,1), mais un institut peut compter plusieurs étudiants (1,N).

Annonce-Mission : une annonce est liée à une seule mission (1,1), mais une mission peut être associée à plusieurs annonces (1,N).

Convention-Institut : un institut peut être lié à plusieurs conventions (1,N), et chaque convention est associée à un seul institut (1,1).

Modèle Logique de Données (MLD)

Schéma Relationnel

Dans cette section, chaque entité du MCD est traduite en une table avec des colonnes spécifiques pour chaque attribut. Les relations entre les entités sont exprimées à travers les clés primaires et étrangères.

1. **Table Candidats :**
    - *IdCandidat* (PK) : Identifiant unique du candidat.
    - *Nom* : Nom du candidat.
    - *Prénom* : Prénom du candidat.
    - *Age* : Âge du candidat.
    - *SituationFamiliale* : Situation familiale du candidat (ex : Célibataire, Marié, etc.).
    - *Mobilite* : Mobilité géographique (Mobile, Non mobile).
    - *SituationProfessionnelle* : Situation actuelle du candidat (En recherche, Salarié, Étudiant).
2. **Table Entreprises :**
    - *IdEntreprise* (PK) : Identifiant unique de l'entreprise.
    - *NomEntreprise* : Nom de l'entreprise.

- *Activite* : Secteur d'activité de l'entreprise (Informatique, Santé, etc.).
- *AdresseEntreprise* : Adresse de l'entreprise.
- *EstCliente* : Indique si l'entreprise est cliente du cabinet (O/N).

3. **Table Missions :**
   - *IdMission* (PK) : Identifiant unique de la mission.
   - *Remuneration* : Rémunération associée à la mission.
   - *DateOuverture* : Date d'ouverture de la mission.
   - *DescriptionMission* : Description de la mission (ex : Développement logiciel, Analyse de données).
   - *EtatMission* : Statut de la mission (Ouverte, Fermée).
   - *IdEntreprise* (FK) : Référence à l'entreprise qui a confié la mission.
4. **Table Fonctions :**
   - *IdFonction* (PK) : Identifiant unique de la fonction.
   - *IntituleFonction* : Nom de la fonction (ex : Ingénieur Logiciel, Chef de Projet).
5. **Table Diplomes :**
   - *IdDiplome* (PK) : Identifiant unique du diplôme.
   - *NomDiplome* : Nom du diplôme (ex : Baccalauréat Scientifique, Licence en Mathématiques).
   - *NiveauEtude* : Niveau d'étude (Bac, Bac+3, Bac+5, etc.).
6. **Table Formations :**
   - *IdFormation* (PK) : Identifiant unique de la formation.
   - *NomFormation* : Nom de la formation (ex : Licence Informatique, Master Data Science).
   - *NiveauFormation* : Niveau de la formation (Bac+3, Bac+5, etc.).
7. **Table Instituts :**
   - *IdInstitut* (PK) : Identifiant unique de l'institut.
   - *NomInstitut* : Nom de l'institut (ex : Université des Sciences, Ecole Supérieure de Commerce).
   - *AdresseInstitut* : Adresse de l'institut.
   - *ActiviteInstitut* : Type d'activité (Enseignement, Commerce, etc.).
8. **Table Etudiants :**
   - *IdEtudiant* (PK) : Identifiant unique de l'étudiant.
   - *Nom* : Nom de l'étudiant.
   - *Prénom* : Prénom de l'étudiant.
   - *NumeroEtudiant* : Numéro d'étudiant.
   - *Telephone* : Numéro de téléphone de l'étudiant.
   - *IdInstitut* (FK) : Référence à l'institut de l'étudiant.
9. **Table Annonces :**
   - *IdAnnonce* (PK) : Identifiant unique de l'annonce.
   - *IntituleAnnonce* : Description succincte de l'offre (ex : Développeur Junior, Data Scientist).
   - *IdMission* (FK) : Référence à la mission associée à l'annonce.
10. **Table Conventions :**
    - *IdConvention* (PK) : Identifiant unique de la convention.
    - *DateDebut* : Date de début de la convention.
    - *DateFin* : Date de fin de la convention.

- *IdInstitut* (FK) : Référence à l'institut concerné par la convention.

11. **Table Candidatures :**
    - *IdCandidature* (PK) : Identifiant unique de la candidature.
    - *DateCandidature* : Date à laquelle la candidature a été soumise.
    - *StatutCandidature* : Statut de la candidature (En cours, Acceptée, Refusée).
    - *CommentaireCandidature* : Commentaire concernant la candidature.
    - *IdCandidat* (FK) : Référence au candidat ayant soumis la candidature.
    - *IdAnnonce* (FK) : Référence à l'annonce à laquelle la candidature correspond.

Contraintes d'intégrité et règles de gestion :

- Chaque entité dispose d'une clé primaire (PK) unique pour garantir l'individualité des enregistrements dans chaque table.
- Des clés étrangères (FK) sont utilisées pour maintenir l'intégrité référentielle entre les tables (ex. : chaque mission est liée à une entreprise, chaque étudiant est lié à un institut).
- Les relations entre les entités sont représentées dans les tables sous forme de clés étrangères pour garantir que les données sont cohérentes et correctement liées.
- Les attributs comme *DateOuverture* dans la table Missions ou *DateCandidature* dans la table Candidatures sont contraints à être des dates valides pour éviter les incohérences.

Scénarios d'Utilisation SQL

Requêtes de Base

1. Sélection des candidats par entreprise

Cette requête permet de récupérer les candidats qui sont associés à une entreprise donnée. Par exemple, pour obtenir tous les candidats travaillant pour "TechCorp", on effectue une jointure entre les tables des candidats et des entreprises, et on applique un filtre pour l'entreprise concernée.

```
SELECT C.NomCandidat, C.PrenomCandidat, C.SituationProfessionnelleCandidat

FROM Candidats C

JOIN Candidats_Entreprises CE ON C.IdCandidat = CE.IdCandidat

JOIN Entreprises E ON CE.IdEntreprise = E.IdEntreprise

WHERE E.NomEntreprise = 'TechCorp';
```

Ici, la relation de base lie les candidats aux entreprises via la table de jointure Candidats_Entreprises. La requête sélectionne les candidats dont l'entreprise associée est "TechCorp".

2. Sélection des missions ouvertes

Cette requête permet de lister toutes les missions qui sont actuellement ouvertes dans le système. On sélectionne les missions où l'état est "Ouverte".

```
SELECT M.DescriptionMission, M.Remuneration

FROM Missions M

WHERE M.EtatMission = 'Ouverte';
```

Cette relation de base est relativement simple : elle sélectionne toutes les missions dont l'état est "Ouverte". Ici, on n’a pas de jointure, juste une condition de filtre sur la table Missions.

3. Sélection des formations suivies par un candidat

Cette requête sélectionne toutes les formations associées à un candidat donné, en utilisant une jointure entre les tables Candidats et Formations via la table Candidats_Formations.

```
SELECT F.IntituleFormation

FROM Formations F

JOIN Candidats_Diplomes CD ON F.IdFormation = CD.IdDiplome

JOIN Candidats C ON CD.IdCandidat = C.IdCandidat

WHERE C.IdCandidat = 1;
```

Dans cette relation de base, on lie les candidats aux formations qu'ils ont suivies via une jointure sur la table de liaison Candidats_Formations. La requête sélectionne les formations pour le candidat dont l'IdCandidat est 1.

4. Sélection des entreprises avec un nombre de candidats associé

Cette requête permet de compter le nombre de candidats associés à chaque entreprise. L'agrégation par COUNT nous permet de savoir combien de candidats travaillent pour chaque entreprise.

```
SELECT E.NomEntreprise, COUNT(C.IdCandidat) AS NombreCandidats
FROM Entreprises E
LEFT JOIN Candidats_Entreprises CE ON E.IdEntreprise = CE.IdEntreprise
```

```
LEFT JOIN Candidats C ON CE.IdCandidat = C.IdCandidat
GROUP BY E.NomEntreprise;
```

Cette relation de base utilise des jointures entre les tables Entreprises, Candidats_Entreprises, et Candidats, et effectue un comptage du nombre de candidats par entreprise.

Fonctions Spécifiques

Introduction

Dans cette section, nous détaillerons les procédures, triggers et fonctions spécifiques qui ont été créés pour répondre aux besoins de gestion des candidats, des missions, et des conventions dans le système. Ces fonctionnalités permettent de gérer automatiquement certaines opérations courantes et de garantir l'intégrité des données.

Présentation des procédures et fonctions

1. **Trigger tr_retrait_candidat**

- **Objectif** : Ce trigger est déclenché après une mise à jour de la table Candidats. Il supprime automatiquement un candidat dont la situation professionnelle est "Recruté" ou "Retiré".
- **Fonctionnement** : Lorsqu'un candidat passe dans l'un de ces états, l'enregistrement correspondant est supprimé.
- **Utilité** : Cela permet de maintenir une table Candidats propre et à jour sans intervention manuelle.

```
CREATE OR REPLACE TRIGGER tr_retrait_candidat
AFTER UPDATE ON Candidats
FOR EACH ROW
BEGIN
  IF :NEW.SituationProfessionnelleCandidat IN ('Recruté', 'Retiré') THEN
    DELETE FROM Candidats WHERE IdCandidat = :NEW.IdCandidat;
  END IF;
END;
```

2. **Fonction assigner_candidats**

- **Objectif** : Cette fonction retourne un curseur contenant les candidats sans emploi qui peuvent être associés à une mission donnée.
- **Fonctionnement** : Un curseur est ouvert avec une requête qui sélectionne les candidats en fonction de leur diplôme et de la mission à pourvoir.
- **Utilité** : Elle permet de récupérer rapidement les candidats éligibles pour une mission en particulier.

```
CREATE OR REPLACE FUNCTION assigner_candidats(p_id_mission IN NUMBER)
```

```
RETURN SYS_REFCURSOR
IS
  candidats_cursor SYS_REFCURSOR;
BEGIN
  OPEN candidats_cursor FOR
    SELECT c.IdCandidat, c.NomCandidat, c.PrenomCandidat
    FROM Candidats c
    JOIN Candidats_Diplomes cd ON c.IdCandidat = cd.IdCandidat
    JOIN Diplomes_Fonctions df ON cd.IdDiplome = df.IdDiplome
    JOIN Fonctions_Missions fm ON df.IdFonction = fm.IdFonction
    JOIN Missions m ON fm.IdMission = m.IdMission
    WHERE m.IdMission = p_id_mission
    AND c.SituationProfessionnelleCandidat = 'Sans emploi';
  RETURN candidats_cursor;
END;
```

3. **Procédure contacter_candidat**

- **Objectif** : Cette procédure enregistre le contact avec un candidat pour une mission donnée.
- **Fonctionnement** : Lorsqu'un candidat est contacté pour une mission, une entrée est ajoutée dans la table Contacts avec la date de contact.
- **Utilité** : Cela permet de garder une trace des interactions avec les candidats.

```
CREATE OR REPLACE PROCEDURE contacter_candidat(p_id_candidat IN NUMBER,
p_id_mission IN NUMBER)
IS
  v_date_contact DATE := SYSDATE;
BEGIN
  INSERT INTO Contacts (IdCandidat, IdMission, DateContact)
  VALUES (p_id_candidat, p_id_mission, v_date_contact);
END;
```

4. **Procédure traiter_candidature_spontanee**

- **Objectif** : Permet de traiter une candidature spontanée en vérifiant si l'entreprise est cliente.
- **Fonctionnement** : Si l'entreprise est cliente, la candidature est enregistrée dans la table Candidatures et une relation est établie entre le candidat et l'entreprise. Si l'entreprise n'est pas cliente, une erreur est levée.
- **Utilité** : Ce processus garantit que seules les candidatures provenant d'entreprises clientes sont traitées.

```
CREATE OR REPLACE PROCEDURE traiter_candidature_spontanee(p_IdCandidat IN
NUMBER, p_IdEntreprise IN NUMBER) AS
BEGIN
  DECLARE
```

```
    v_est_cliente CHAR(1);
  BEGIN
    SELECT EstCliente INTO v_est_cliente
    FROM Entreprises
    WHERE IdEntreprise = p_IdEntreprise;

    IF v_est_cliente = 'Y' THEN
      INSERT INTO Candidatures (IdCandidature, DateCandidature, StatutCandidature)
      VALUES (seq_candidatures.NEXTVAL, SYSDATE, 'En cours');

      INSERT INTO Candidats_Entreprises (IdCandidat, IdEntreprise)
      VALUES (p_IdCandidat, p_IdEntreprise);
    ELSE
      RAISE_APPLICATION_ERROR(-20001, 'L''entreprise n''est pas cliente');
    END IF;
  END;
END traiter_candidature_spontanee;
```

5. **Procédure ajouter_etudiant_candidat**

- **Objectif** : Cette procédure permet d'ajouter un étudiant en tant que candidat dans la table Candidats.
- **Fonctionnement** : Un étudiant est copié depuis la table Etudiants vers la table Candidats avec des valeurs par défaut pour certains champs.
- **Utilité** : Cela permet de facilement transformer un étudiant en candidat lorsqu'il termine sa formation.

```
CREATE OR REPLACE PROCEDURE ajouter_etudiant_candidat(p_id_etudiant IN
NUMBER)
IS
BEGIN
  INSERT INTO Candidats (IdCandidat, NomCandidat, PrenomCandidat,
SituationFamilialeCandidat, SituationProfessionnelleCandidat, MobiliteCandidat)
  SELECT IdEtudiant, NomEtudiant, PrenomEtudiant, '', 'Sans emploi', ''
  FROM Etudiants
  WHERE IdEtudiant = p_id_etudiant;
END;
```

6. **Procédure terminer_mission**

- **Objectif** : Supprime une mission si elle est terminée ou annulée.
- **Fonctionnement** : Après une vérification de l'état de la mission, celle-ci est supprimée si elle est terminée ou annulée.
- **Utilité** : Cette procédure aide à maintenir la base de données à jour en supprimant les missions qui ne sont plus actives.

```
CREATE OR REPLACE PROCEDURE terminer_mission(p_id_mission IN NUMBER)
```

```
IS
BEGIN
  DELETE FROM Missions WHERE IdMission = p_id_mission
  AND (EtatMission = 'Terminé' OR EtatMission = 'Annulé');
END;
```

Trigger trg_supprimer_mission

- **Objectif** : Ce trigger est déclenché après la mise à jour de l'état d'une mission. Si l'état de la mission devient "Terminée", la procédure supprimer_mission est appelée.
- **Utilité** : Cela permet d'automatiser la suppression des missions terminées sans intervention manuelle.

```
CREATE OR REPLACE TRIGGER trg_supprimer_mission
AFTER UPDATE OF EtatMission ON Missions
FOR EACH ROW
BEGIN
  IF :NEW.EtatMission = 'Terminée' THEN
    terminer_mission(:NEW.IdMission);
  END IF;
END trg_supprimer_mission;
```

Maintenabilité et Suppression de Données

Introduction :

La maintenabilité et la suppression de données sont des aspects cruciaux pour garantir le bon fonctionnement d'un système de gestion de base de données sur le long terme. Une gestion efficace des données permet non seulement de libérer de l'espace de stockage, mais aussi de préserver la performance des requêtes et d'assurer la conformité avec les politiques de rétention des données. Cette section présente les mécanismes mis en place pour garantir une gestion durable des données dans le système.

Stratégies de maintenabilité des données

1. **Indexation des tables**
    - **Objectif** : L'utilisation d'index permet d'optimiser les performances des requêtes en réduisant le temps d'accès aux données.
    - **Mise en œuvre** : Des index ont été créés sur les colonnes les plus utilisées dans les filtres ou les jointures, comme IdCandidat, IdMission, IdEntreprise, et DateCandidature.
    - **Avantages** : Cette stratégie réduit le coût des recherches, des mises à jour et des suppressions dans les tables fréquemment utilisées.

2. **Normalisation des données**
    - **Objectif** : La normalisation des données a été réalisée pour éviter les redondances et garantir l'intégrité des données.
    - **Mise en œuvre** : Le modèle relationnel est conçu selon les principes de la 3NF (Troisième forme normale), ce qui permet d'éviter la duplication des données et d'assurer leur cohérence.
    - **Avantages** : Cela améliore la maintenabilité en facilitant les mises à jour et les suppressions, et en réduisant le risque d'anomalies dans les données.
3. **Utilisation des contraintes d'intégrité**
    - **Objectif** : Les contraintes d'intégrité (clés primaires, clés étrangères, contraintes de vérification) sont utilisées pour assurer que seules des données valides et cohérentes sont insérées ou modifiées dans les tables.
    - **Mise en œuvre** : Des clés primaires et étrangères ont été définies sur les tables pertinentes (par exemple, Candidats(IdCandidat), Missions(IdMission), etc.), et des contraintes de vérification sont appliquées pour limiter les erreurs de saisie.
4. **Archivage ou suppression des données obsolètes**
    - **Objectif** : Lorsque certaines données ne sont plus nécessaires mais doivent être conservées pour des raisons légales ou analytiques, elles peuvent être archivées.
    - **Mise en œuvre** : Un processus d'archivage automatique ou manuel est mis en place pour déplacer les anciennes données vers des tables d'archivage, par exemple, des missions ou des candidatures clôturées depuis plus de deux ans.
    - **Avantages** : Cela permet de libérer de l'espace de stockage tout en respectant la réglementation concernant la conservation des données.
5. **Suppression des données obsolètes ou inutiles**
    - **Objectif** : Il est important de supprimer les données qui ne sont plus nécessaires, comme les candidatures rejetées ou les missions annulées.
    - **Mise en œuvre** : Des procédures de suppression sont mises en place pour nettoyer régulièrement la base de données, tout en respectant les contraintes d'intégrité.

- **Avantages** : La suppression de données obsolètes améliore la performance de la base de données et garantit la conformité avec les politiques internes ou les réglementations en matière de rétention des données.

```
CREATE OR REPLACE PROCEDURE supprimer_donnees_obsoletes IS
BEGIN
  DELETE FROM Candidatures WHERE DateCandidature < SYSDATE - INTERVAL '2' YEAR AND StatutCandidature = 'Refusé';
  DELETE FROM Missions WHERE DateCloture < SYSDATE - INTERVAL '2' YEAR AND EtatMission = 'Annulé';
END;
```

6. Suppression des données après retrait d'un candidat
    - **Objectif** : Lorsque la situation professionnelle d'un candidat est mise à jour pour indiquer un départ ou une inactivité, les données associées peuvent être supprimées.
    - **Mise en œuvre** : Un trigger est utilisé pour automatiser la suppression des données liées à un candidat lorsqu'il change de statut, par exemple lorsqu'il est recruté ou retiré.
    - **Avantages** : Cela évite l'accumulation de données obsolètes et assure que la base de données reste à jour sans intervention manuelle.

```
CREATE OR REPLACE TRIGGER trg_supprimer_candidat
AFTER UPDATE OF SituationProfessionnelleCandidat ON Candidats
FOR EACH ROW
BEGIN
  IF :NEW.SituationProfessionnelleCandidat IN ('Recruté', 'Retiré') THEN
    DELETE FROM Candidats WHERE IdCandidat = :NEW.IdCandidat;
  END IF;
END;
```

Conclusion

Ce projet de conception et de gestion d'une base de données pour le cabinet de conseil en ressources humaines a permis de structurer et d'organiser efficacement les données liées aux candidatures, aux missions, et aux formations des candidats. À travers la création du Modèle Conceptuel de Données (MCD), du Modèle Logique de Données (MLD), ainsi que des scénarios d'utilisation SQL, nous avons mis en place une base solide pour gérer les flux d'information tout au long du processus de recrutement.

Les différentes stratégies mises en œuvre, notamment la normalisation des données, l'utilisation des contraintes d'intégrité et l'indexation, assurent non seulement la cohérence et l'efficacité du système, mais également sa performance à long terme. La partie consacrée à la maintenabilité et à la suppression des données a mis en lumière l'importance de la gestion durable des informations, en garantissant la réactivité du système tout en respectant les contraintes de rétention légales et organisationnelles.

Les requêtes complexes et l'automatisation des processus, comme l'archivage et la suppression des données obsolètes, ont été conçues pour minimiser l'intervention manuelle et optimiser la gestion des données. L'utilisation de procédures stockées, de triggers et de mécanismes de suppression en cascade permet de maintenir l'intégrité et la performance de la base de données, tout en assurant sa pérennité.

En conclusion, ce projet répond à la problématique de gestion efficace des données dans un environnement dynamique, en garantissant une expérience utilisateur fluide, une gestion optimale des candidatures et des missions, et une conformité stricte avec les exigences légales. Il constitue une base solide pour toute évolution future du système, et peut être facilement adapté à des besoins spécifiques de l'entreprise.

Annexes

Exemples de Requêtes SQL

Liste de tous les candidats et leurs diplomes :

```
SELECT C.NomCandidat, C.PrenomCandidat, D.IntituleDiplome AS Diplôme

FROM Candidats C

JOIN Candidats_Diplomes CD ON C.IdCandidat = CD.IdCandidat

JOIN Diplomes D ON CD.IdDiplome = D.IdDiplome;
```

Trouver les entreprises et les missions qu'elles offrent :

```
SELECT

  E.NomEntreprise,

  E.ActiviteEntreprise,

  M.DescriptionMission,
```

```
    M.Remuneration,
    M.DateOuverture,
    M.EtatMission
FROM
    Entreprises E
JOIN
    Candidats_Entreprises CE ON E.IdEntreprise = CE.IdEntreprise
JOIN
    Candidats C ON CE.IdCandidat = C.IdCandidat
JOIN
    Candidatures CA ON C.IdCandidat = CA.IdCandidature
JOIN
    Requiert R ON CA.IdCandidature = R.IdCandidature
JOIN
    Missions M ON R.IdMission = M.IdMission;
```

Rechercher les candidatures en cours pour une mission spécifique :

```
SELECT
    C.NomCandidat,
    C.PrenomCandidat,
    CA.DateCandidature,
    CA.StatutCandidature,
    CA.CommentaireRH
FROM
    Candidatures CA
JOIN
    Requiert R ON CA.IdCandidature = R.IdCandidature
JOIN
    Candidate CN ON CA.IdCandidature = CN.IdCandidature
JOIN
    Candidats C ON CN.IdCandidat = C.IdCandidat
WHERE
    R.IdMission = 1
    AND CA.StatutCandidature = 'En cours';
```

Lister les étudiants et les formations qu'ils suivent :

```
SELECT E.NomEtudiant, E.PrenomEtudiant, F.IntituleFormation, I.NomInstitut
```

```
FROM Etudiants E
JOIN Etudiants_Instituts EI ON E.IdEtudiant = EI.IdEtudiant
JOIN Formations_Instituts FI ON EI.IdInstitut = FI.IdInstitut
JOIN Formations F ON FI.IdFormation = F.IdFormation
JOIN Instituts I ON EI.IdInstitut = I.IdInstitut;
```

Afficher les entreprises partenaires d'un institut (qui prennent un étudiant en stage ou emploi) :

```
SELECT DISTINCT e.NomEntreprise, e.ActiviteEntreprise, e.AdresseEntreprise
FROM Entreprises e
JOIN Conventions_Entreprises ce ON e.IdEntreprise = ce.IdEntreprise
JOIN Conventions c ON ce.IdConvention = c.IdConvention
JOIN Conventions_Etudiants cee ON c.IdConvention = cee.IdConvention
JOIN Etudiants_Instituts ei ON cee.IdEtudiant = ei.IdEtudiant
JOIN Instituts i ON ei.IdInstitut = i.IdInstitut
WHERE i.IdInstitut = 1;
```

Trouver les candidats ayant postulé à une annonce spécifique :

```
SELECT c.NomCandidat, c.PrenomCandidat, c.MailCandidat
FROM Candidats c
JOIN Candidate ca ON c.IdCandidat = ca.IdCandidat
JOIN Candidatures cd ON ca.IdCandidature = cd.IdCandidature
JOIN Requiert r ON cd.IdCandidature = r.IdCandidature
JOIN Missions m ON r.IdMission = m.IdMission
JOIN Annonces_Missions am ON m.IdMission = am.IdMission
JOIN Annonces a ON am.IdAnnonce = a.IdAnnonce
WHERE a.IdAnnonce = 1;
```

Rechercher les missions et leur entreprise associée pour un candidat :

```
SELECT m.DescriptionMission, e.NomEntreprise, e.ActiviteEntreprise, e.AdresseEntreprise
FROM Candidats c
JOIN Candidate ca ON c.IdCandidat = ca.IdCandidat
JOIN Candidatures cd ON ca.IdCandidature = cd.IdCandidature
JOIN Requiert r ON cd.IdCandidature = r.IdCandidature
JOIN Missions m ON r.IdMission = m.IdMission
JOIN Annonces_Missions am ON m.IdMission = am.IdMission
JOIN Annonces a ON am.IdAnnonce = a.IdAnnonce
JOIN Annonces_Instituts ai ON a.IdAnnonce = ai.IdAnnonce
JOIN Instituts i ON ai.IdInstitut = i.IdInstitut
JOIN Etudiants_Instituts ei ON i.IdInstitut = ei.IdInstitut
```

```
JOIN Etudiants e2 ON ei.IdEtudiant = e2.IdEtudiant
JOIN Candidats_Entreprises ce ON c.IdCandidat = ce.IdCandidat
JOIN Entreprises e ON ce.IdEntreprise = e.IdEntreprise
WHERE c.IdCandidat = :id_candidat;  1
```

Trouver les conventions liées à un étudiant :

```
SELECT c.*
FROM Conventions c
JOIN Conventions_Etudiants ce ON c.idConvention = ce.idConvention
JOIN Etudiants e ON ce.idEtudiant = e.idEtudiant
WHERE e.idEtudiant = :idEtudiant;
```

Scipts de créations

Créations des entités et relations :

```
-- Création des tables principales

CREATE TABLE Diplomes (
  IdDiplome NUMBER PRIMARY KEY,
  IntituleDiplome VARCHAR2(100),
  NiveauEtudeDiplome VARCHAR2(50)
);

CREATE TABLE Fonctions (
  IdFonction NUMBER PRIMARY KEY,
  DescriptionFonction VARCHAR2(255)
);

CREATE TABLE Candidats (
  IdCandidat NUMBER PRIMARY KEY,
  NomCandidat VARCHAR2(50),
  PrenomCandidat VARCHAR2(50),
  AgeCandidat NUMBER,
  MailCandidat VARCHAR2(100),
  SituationFamilialeCandidat VARCHAR2(50),
  MobiliteCandidat VARCHAR2(50),
  TelephoneCandidat VARCHAR2(20),
  AdresseCandidat VARCHAR2(255),
```

```
    SituationProfessionnelleCandidat VARCHAR2(50)
);

CREATE TABLE Entreprises (
    IdEntreprise NUMBER PRIMARY KEY,
    NomEntreprise VARCHAR2(100),
    ActiviteEntreprise VARCHAR2(100),
    AdresseEntreprise VARCHAR2(255),
    EstCliente CHAR(1) CHECK (EstCliente IN ('Y', 'N')),
    SIRETEntreprise VARCHAR2(14),
    NumeroTelephoneEntreprise VARCHAR2(20),
    EmailEntreprise VARCHAR2(100)
);

CREATE TABLE Instituts (
    IdInstitut NUMBER PRIMARY KEY,
    SIRETInstitut VARCHAR2(14),
    NomInstitut VARCHAR2(100),
    AdresseInstitut VARCHAR2(255),
    ActiviteInstitut VARCHAR2(100)
);

CREATE TABLE Formations (
    IdFormation NUMBER PRIMARY KEY,
    IntituleFormation VARCHAR2(100),
    NiveauEtude VARCHAR2(50),
    Description VARCHAR2(255)
);

CREATE TABLE Etudiants (
    IdEtudiant NUMBER PRIMARY KEY,
    NEtudiant VARCHAR2(20),
    NomEtudiant VARCHAR2(50),
    PrenomEtudiant VARCHAR2(50),
    TelephoneEtudiant VARCHAR2(20),
    AdresseEtudiant VARCHAR2(255),
```

```
  MailEtudiant VARCHAR2(100)
);

CREATE TABLE Annonces (
  IdAnnonce NUMBER PRIMARY KEY,
  IntituleAnnonce VARCHAR2(100),
  DescriptionAnnonce VARCHAR2(255)
);

CREATE TABLE Missions (
  IdMission NUMBER PRIMARY KEY,
  Remuneration NUMBER,
  DateOuverture DATE,
  DescriptionMission VARCHAR2(255),
  EtatMission VARCHAR2(50)
);

CREATE TABLE Conventions (
  IdConvention NUMBER PRIMARY KEY,
  DateDebutConvention DATE,
  DateFinConvention DATE
);

CREATE TABLE Candidatures (
  IdCandidature NUMBER PRIMARY KEY,
  DateCandidature DATE,
  StatutCandidature VARCHAR2(50),
  CommentaireRH VARCHAR2(255)
);

CREATE TABLE Contacts (
  IdContact NUMBER PRIMARY KEY,
  IdCandidat NUMBER,
  IdMission NUMBER,
  DateContact DATE,
  FOREIGN KEY (IdCandidat) REFERENCES Candidats(IdCandidat),
```

```
  FOREIGN KEY (IdMission) REFERENCES Missions(IdMission)
);

-- Création des tables de liaison

CREATE TABLE Diplomes_Fonctions (
  IdDiplome NUMBER,
  IdFonction NUMBER,
  PRIMARY KEY (IdDiplome, IdFonction),
  FOREIGN KEY (IdDiplome) REFERENCES Diplomes(IdDiplome),
  FOREIGN KEY (IdFonction) REFERENCES Fonctions(IdFonction)
);

CREATE TABLE Candidats_Diplomes (
  IdCandidat NUMBER,
  IdDiplome NUMBER,
  PRIMARY KEY (IdCandidat, IdDiplome),
  FOREIGN KEY (IdCandidat) REFERENCES Candidats(IdCandidat),
  FOREIGN KEY (IdDiplome) REFERENCES Diplomes(IdDiplome)
);

CREATE TABLE Aptitudes_Candidats (
  IdAptitude NUMBER PRIMARY KEY,
  IdCandidat NUMBER,
  FOREIGN KEY (IdCandidat) REFERENCES Candidats(IdCandidat)
);

CREATE TABLE Candidats_Entreprises (
  IdCandidat NUMBER,
  IdEntreprise NUMBER,
  PRIMARY KEY (IdCandidat, IdEntreprise),
  FOREIGN KEY (IdCandidat) REFERENCES Candidats(IdCandidat),
  FOREIGN KEY (IdEntreprise) REFERENCES Entreprises(IdEntreprise)
);
CREATE TABLE Candidats_Contacts (
  IdCandidat NUMBER,
```

```
    IdContact NUMBER,
    PRIMARY KEY (IdCandidat, IdContact),
    FOREIGN KEY (IdCandidat) REFERENCES Candidats(IdCandidat),
    FOREIGN KEY (IdContact) REFERENCES Contacts(IdContact)
);

CREATE TABLE Conventions_Entreprises (
    IdConvention NUMBER,
    IdEntreprise NUMBER,
    PRIMARY KEY (IdConvention, IdEntreprise),
    FOREIGN KEY (IdConvention) REFERENCES Conventions(IdConvention),
    FOREIGN KEY (IdEntreprise) REFERENCES Entreprises(IdEntreprise)
);

CREATE TABLE Conventions_Missions (
    IdConvention NUMBER,
    IdMission NUMBER,
    PRIMARY KEY (IdConvention, IdMission),
    FOREIGN KEY (IdConvention) REFERENCES Conventions(IdConvention),
    FOREIGN KEY (IdMission) REFERENCES Missions(IdMission)
);

CREATE TABLE Conventions_Etudiants (
    IdConvention NUMBER,
    IdEtudiant NUMBER,
    PRIMARY KEY (IdConvention, IdEtudiant),
    FOREIGN KEY (IdConvention) REFERENCES Conventions(IdConvention),
    FOREIGN KEY (IdEtudiant) REFERENCES Etudiants(IdEtudiant)
);

CREATE TABLE Formations_Instituts (
    IdFormation NUMBER,
    IdInstitut NUMBER,
    PRIMARY KEY (IdFormation, IdInstitut),
    FOREIGN KEY (IdFormation) REFERENCES Formations(IdFormation),
    FOREIGN KEY (IdInstitut) REFERENCES Instituts(IdInstitut)
```

```
);

CREATE TABLE Etudiants_Instituts (
  IdEtudiant NUMBER,
  IdInstitut NUMBER,
  PRIMARY KEY (IdEtudiant, IdInstitut),
  FOREIGN KEY (IdEtudiant) REFERENCES Etudiants(IdEtudiant),
  FOREIGN KEY (IdInstitut) REFERENCES Instituts(IdInstitut)
);

CREATE TABLE Annonces_Missions (
  IdAnnonce NUMBER,
  IdMission NUMBER,
  PRIMARY KEY (IdAnnonce, IdMission),
  FOREIGN KEY (IdAnnonce) REFERENCES Annonces(IdAnnonce),
  FOREIGN KEY (IdMission) REFERENCES Missions(IdMission)
);

CREATE TABLE Annonces_Instituts (
  IdAnnonce NUMBER,
  IdInstitut NUMBER,
  PRIMARY KEY (IdAnnonce, IdInstitut),
  FOREIGN KEY (IdAnnonce) REFERENCES Annonces(IdAnnonce),
  FOREIGN KEY (IdInstitut) REFERENCES Instituts(IdInstitut)
);

CREATE TABLE Fonctions_Missions (
  IdFonction NUMBER,
  IdMission NUMBER,
  PRIMARY KEY (IdFonction, IdMission),
  FOREIGN KEY (IdFonction) REFERENCES Fonctions(IdFonction),
  FOREIGN KEY (IdMission) REFERENCES Missions(IdMission)
);

-- Création des relations pour les candidatures
```

```
CREATE TABLE Requiert (
   IdMission NUMBER,
   IdCandidature NUMBER,
   PRIMARY KEY (IdMission, IdCandidature),
   FOREIGN KEY (IdMission) REFERENCES Missions(IdMission),
   FOREIGN KEY (IdCandidature) REFERENCES Candidatures(IdCandidature)
);

CREATE TABLE Candidate (
   IdCandidat NUMBER,
   IdCandidature NUMBER,
   PRIMARY KEY (IdCandidat, IdCandidature),
   FOREIGN KEY (IdCandidat) REFERENCES Candidats(IdCandidat),
   FOREIGN KEY (IdCandidature) REFERENCES Candidatures(IdCandidature)
);
```

Insertions des données :

```
-- Insertion des Diplomes
INSERT INTO Diplomes (IdDiplome, IntituleDiplome, NiveauEtudeDiplome) VALUES (1, 'Baccalauréat Scientifique', 'Bac');
INSERT INTO Diplomes (IdDiplome, IntituleDiplome, NiveauEtudeDiplome) VALUES (2, 'Licence en Mathématiques', 'Bac+3');
INSERT INTO Diplomes (IdDiplome, IntituleDiplome, NiveauEtudeDiplome) VALUES (3, 'Master en Informatique', 'Bac+5');

-- Insertion des Fonctions
INSERT INTO Fonctions (IdFonction, DescriptionFonction) VALUES (1, 'Ingénieur Logiciel');
INSERT INTO Fonctions (IdFonction, DescriptionFonction) VALUES (2, 'Data Scientist');
INSERT INTO Fonctions (IdFonction, DescriptionFonction) VALUES (3, 'Chef de Projet');

-- Insertion des Candidats
INSERT INTO Candidats (IdCandidat, NomCandidat, PrenomCandidat, AgeCandidat, MailCandidat, SituationFamilialeCandidat,
MobiliteCandidat, TelephoneCandidat, AdresseCandidat, SituationProfessionnelleCandidat)
VALUES (1, 'Bernard', 'Luc', 25, 'luc.bernard@example.com', 'Célibataire', 'Mobile', '0712345678', '1 Rue de la Paix', 'En recherche');
INSERT INTO Candidats (IdCandidat, NomCandidat, PrenomCandidat, AgeCandidat, MailCandidat, SituationFamilialeCandidat,
MobiliteCandidat, TelephoneCandidat, AdresseCandidat, SituationProfessionnelleCandidat)
VALUES (2, 'Petit', 'Chloé', 30, 'chloe.petit@example.com', 'Marié(e)', 'Non mobile', '0723456789', '2 Avenue du Roi', 'Salarié');
INSERT INTO Candidats (IdCandidat, NomCandidat, PrenomCandidat, AgeCandidat, MailCandidat, SituationFamilialeCandidat,
MobiliteCandidat, TelephoneCandidat, AdresseCandidat, SituationProfessionnelleCandidat)
VALUES (3, 'Moreau', 'Lucas', 27, 'lucas.moreau@example.com', 'Célibataire', 'Mobile', '0734567890', '3 Boulevard de l''Amiral',
'Étudiant');
```

```
-- Insertion des Entreprises

INSERT INTO Entreprises (IdEntreprise, NomEntreprise, ActiviteEntreprise, AdresseEntreprise, EstCliente, SIRETEntreprise, NumeroTelephoneEntreprise, EmailEntreprise)

VALUES (1, 'TechCorp', 'Informatique', '5 Rue de l''Innovation', 'Y', '45678901234567', '0812345678', 'contact@techcorp.com');

INSERT INTO Entreprises (IdEntreprise, NomEntreprise, ActiviteEntreprise, AdresseEntreprise, EstCliente, SIRETEntreprise, NumeroTelephoneEntreprise, EmailEntreprise)

VALUES (2, 'MediHealth', 'Santé', '10 Avenue de la Santé', 'N', '56789012345678', '0823456789', 'info@medihealth.com');


-- Insertion des Instituts

INSERT INTO Instituts (IdInstitut, SIRETInstitut, NomInstitut, AdresseInstitut, ActiviteInstitut)

VALUES (1, '12345678901234', 'Institut Polytechnique', '1 Rue de l''Ingénieur', 'Enseignement');

INSERT INTO Instituts (IdInstitut, SIRETInstitut, NomInstitut, AdresseInstitut, ActiviteInstitut)

VALUES (2, '23456789012345', 'Université des Sciences', '12 Avenue de la Recherche', 'Enseignement');

INSERT INTO Instituts (IdInstitut, SIRETInstitut, NomInstitut, AdresseInstitut, ActiviteInstitut)

VALUES (3, '34567890123456', 'Ecole Supérieure de Commerce', '34 Boulevard des Affaires', 'Commerce');


-- Insertion des Formations

INSERT INTO Formations (IdFormation, IntituleFormation, NiveauEtude, Description)

VALUES (1, 'Licence Informatique', 'Bac+3', 'Formation en informatique générale');

INSERT INTO Formations (IdFormation, IntituleFormation, NiveauEtude, Description)

VALUES (2, 'Master Data Science', 'Bac+5', 'Formation en science des données');

INSERT INTO Formations (IdFormation, IntituleFormation, NiveauEtude, Description)

VALUES (3, 'Bachelor Commerce', 'Bac+3', 'Formation en commerce international');


-- Insertion des Etudiants

INSERT INTO Etudiants (IdEtudiant, NEtudiant, NomEtudiant, PrenomEtudiant, TelephoneEtudiant, AdresseEtudiant, MailEtudiant)

VALUES (1, 'ET001', 'Dupont', 'Jean', '0612345678', '1 Rue de Paris', 'jean.dupont@example.com');

INSERT INTO Etudiants (IdEtudiant, NEtudiant, NomEtudiant, PrenomEtudiant, TelephoneEtudiant, AdresseEtudiant, MailEtudiant)

VALUES (2, 'ET002', 'Martin', 'Marie', '0623456789', '2 Avenue des Champs', 'marie.martin@example.com');

INSERT INTO Etudiants (IdEtudiant, NEtudiant, NomEtudiant, PrenomEtudiant, TelephoneEtudiant, AdresseEtudiant, MailEtudiant)

VALUES (3, 'ET003', 'Nguyen', 'Thierry', '0634567890', '3 Boulevard de la Liberté', 'thierry.nguyen@example.com');


-- Insertion des Annonces

INSERT INTO Annonces (IdAnnonce, IntituleAnnonce, DescriptionAnnonce)

VALUES (1, 'Offre Développeur Junior', 'Poste de développeur pour débutants');

INSERT INTO Annonces (IdAnnonce, IntituleAnnonce, DescriptionAnnonce)
```

```
VALUES (2, 'Offre Data Scientist', 'Poste pour expert en data science');

-- Insertion des Missions
INSERT INTO Missions (IdMission, Remuneration, DateOuverture, DescriptionMission, EtatMission)
VALUES (1, 3000, TO_DATE('2024-01-01', 'YYYY-MM-DD'), 'Développement logiciel', 'Ouverte');
INSERT INTO Missions (IdMission, Remuneration, DateOuverture, DescriptionMission, EtatMission)
VALUES (2, 4000, TO_DATE('2024-02-01', 'YYYY-MM-DD'), 'Analyse de données', 'Ouverte');

-- Insertion des Conventions
INSERT INTO Conventions (IdConvention, DateDebutConvention, DateFinConvention)
VALUES (1, TO_DATE('2023-01-15', 'YYYY-MM-DD'), TO_DATE('2023-06-15', 'YYYY-MM-DD'));
INSERT INTO Conventions (IdConvention, DateDebutConvention, DateFinConvention)
VALUES (2, TO_DATE('2023-09-01', 'YYYY-MM-DD'), TO_DATE('2024-02-01', 'YYYY-MM-DD'));

-- Insertion d'une prise de contact avec un candidat
INSERT INTO Contacts (IdContact, IdCandidat, IdMission, DateContact)
VALUES (1, 1, 2, SYSDATE);

-- Insertion des Candidatures
INSERT INTO Candidatures (IdCandidature, DateCandidature, StatutCandidature, CommentaireRH)
VALUES (1, TO_DATE('2024-02-15', 'YYYY-MM-DD'), 'En cours', 'Bonne candidature');
INSERT INTO Candidatures (IdCandidature, DateCandidature, StatutCandidature, CommentaireRH)
VALUES (2, TO_DATE('2024-02-16', 'YYYY-MM-DD'), 'Refusé', 'Pas assez d''expérience');

-- Insertion des relations entre les tables

-- Diplômes - Fonctions
INSERT INTO Diplomes_Fonctions (IdDiplome, IdFonction)
VALUES (1, 1);
INSERT INTO Diplomes_Fonctions (IdDiplome, IdFonction)
VALUES (2, 2);
INSERT INTO Diplomes_Fonctions (IdDiplome, IdFonction)
VALUES (3, 3);

-- Candidats - Diplômes
INSERT INTO Candidats_Diplomes (IdCandidat, IdDiplome)
```

```
VALUES (1, 1);

INSERT INTO Candidats_Diplomes (IdCandidat, IdDiplome)

VALUES (1, 2);

INSERT INTO Candidats_Diplomes (IdCandidat, IdDiplome)

VALUES (2, 2);

INSERT INTO Candidats_Diplomes (IdCandidat, IdDiplome)

VALUES (3, 3);


-- Candidats - Entreprises

INSERT INTO Candidats_Entreprises (IdCandidat, IdEntreprise)

VALUES (1, 1);

INSERT INTO Candidats_Entreprises (IdCandidat, IdEntreprise)

VALUES (2, 2);


-- Conventions - Entreprises

INSERT INTO Conventions_Entreprises (IdConvention, IdEntreprise)

VALUES (1, 1);

INSERT INTO Conventions_Entreprises (IdConvention, IdEntreprise)

VALUES (2, 2);


-- Conventions - Etudiants

INSERT INTO Conventions_Etudiants (IdConvention, IdEtudiant)

VALUES (1, 1);

INSERT INTO Conventions_Etudiants (IdConvention, IdEtudiant)

VALUES (2, 2);


-- Annonces - Missions

INSERT INTO Annonces_Missions (IdAnnonce, IdMission)

VALUES (1, 1);

INSERT INTO Annonces_Missions (IdAnnonce, IdMission)

VALUES (2, 2);


-- Annonces - Instituts

INSERT INTO Annonces_Instituts (IdAnnonce, IdInstitut)

VALUES (1, 1);

INSERT INTO Annonces_Instituts (IdAnnonce, IdInstitut)
```

```
VALUES (2, 2);

-- Fonctions - Missions
INSERT INTO Fonctions_Missions (IdFonction, IdMission)
VALUES (1, 1);
INSERT INTO Fonctions_Missions (IdFonction, IdMission)
VALUES (2, 2);

--Etudiants_Instituts
INSERT INTO Etudiants_Instituts (IdEtudiant, IdInstitut)
VALUES (1, 1);

INSERT INTO Etudiants_Instituts (IdEtudiant, IdInstitut)
VALUES (2, 2);

--Formations_Instituts
INSERT INTO Formations_Instituts (IdFormation, IdInstitut)
VALUES (1, 1);

INSERT INTO Formations_Instituts (IdFormation, IdInstitut)
VALUES (2, 2);

--Requiert
INSERT INTO Requiert (IdMission, IdCandidature) VALUES (1, 1);
--Candidate
INSERT INTO Candidate (IdCandidat, IdCandidature) VALUES (1, 1);
INSERT INTO Candidate (IdCandidat, IdCandidature) VALUES (2, 2);
```

Triggers, fonctions et procédures :

```
CREATE OR REPLACE TRIGGER tr_retrait_candidat
AFTER UPDATE ON Candidats
FOR EACH ROW
BEGIN
  IF :NEW.SituationProfessionnelleCandidat IN ('Recruté', 'Retiré') THEN
```

```
    DELETE FROM Candidats WHERE IdCandidat = :NEW.IdCandidat;
  END IF;
END;
/

CREATE OR REPLACE FUNCTION assigner_candidats(p_id_mission IN NUMBER)
RETURN SYS_REFCURSOR
IS
  candidats_cursor SYS_REFCURSOR;
BEGIN
  OPEN candidats_cursor FOR
    SELECT c.IdCandidat, c.NomCandidat, c.PrenomCandidat
    FROM Candidats c
    JOIN Candidats_Diplomes cd ON c.IdCandidat = cd.IdCandidat
    JOIN Diplomes_Fonctions df ON cd.IdDiplome = df.IdDiplome
    JOIN Fonctions_Missions fm ON df.IdFonction = fm.IdFonction
    JOIN Missions m ON fm.IdMission = m.IdMission
    WHERE m.IdMission = p_id_mission
    AND c.SituationProfessionnelleCandidat = 'Sans emploi';
  RETURN candidats_cursor;
END;
/

CREATE OR REPLACE PROCEDURE contacter_candidat(p_id_candidat IN NUMBER, p_id_mission IN NUMBER)
IS
  v_date_contact DATE := SYSDATE;
BEGIN
  INSERT INTO Contacts (IdCandidat, IdMission, DateContact)
  VALUES (p_id_candidat, p_id_mission, v_date_contact);
END;
/

CREATE SEQUENCE seq_candidatures
START WITH 1
INCREMENT BY 1
NOCACHE
```

```
NOCYCLE;



CREATE OR REPLACE PROCEDURE traiter_candidature_spontanee(p_IdCandidat IN NUMBER, p_IdEntreprise IN NUMBER) AS
BEGIN
  DECLARE
    v_est_cliente CHAR(1);
  BEGIN
    SELECT EstCliente INTO v_est_cliente
    FROM Entreprises
    WHERE IdEntreprise = p_IdEntreprise;

    IF v_est_cliente = 'Y' THEN
      INSERT INTO Candidatures (IdCandidature, DateCandidature, StatutCandidature)
      VALUES (seq_candidatures.NEXTVAL, SYSDATE, 'En cours');

      INSERT INTO Candidats_Entreprises (IdCandidat, IdEntreprise)
      VALUES (p_IdCandidat, p_IdEntreprise);
    ELSE
      RAISE_APPLICATION_ERROR(-20001, 'L''entreprise n''est pas cliente');
    END IF;
  END;
END traiter_candidature_spontanee;
/
CREATE OR REPLACE PROCEDURE ajouter_etudiant_candidat(p_id_etudiant IN NUMBER)
IS
BEGIN
  INSERT INTO Candidats (IdCandidat, NomCandidat, PrenomCandidat, SituationFamilialeCandidat,
SituationProfessionnelleCandidat, MobiliteCandidat)
  SELECT IdEtudiant, NomEtudiant, PrenomEtudiant, '', 'Sans emploi', ''
  FROM Etudiants
  WHERE IdEtudiant = p_id_etudiant;
END;
/

CREATE OR REPLACE PROCEDURE terminer_mission(p_id_mission IN NUMBER)
```

```
IS
BEGIN
  DELETE FROM Missions WHERE IdMission = p_id_mission
  AND (EtatMission = 'Terminé' OR EtatMission = 'Annulé');
END;
/

CREATE OR REPLACE PROCEDURE mettre_a_jour_statut_candidature(p_IdCandidature IN NUMBER, p_nouveau_statut IN
VARCHAR2) AS
BEGIN
  UPDATE Candidatures
  SET StatutCandidature = p_nouveau_statut
  WHERE IdCandidature = p_IdCandidature;

  IF p_nouveau_statut = 'Retiré' THEN
    DELETE FROM Candidats_Entreprises
    WHERE IdCandidat = (SELECT IdCandidat FROM Candidatures WHERE IdCandidature = p_IdCandidature);
  END IF;
END mettre_a_jour_statut_candidature;

/
CREATE OR REPLACE TRIGGER trg_supprimer_mission
AFTER UPDATE OF EtatMission ON Missions
FOR EACH ROW
BEGIN
  IF :NEW.EtatMission = 'Terminée' THEN
    terminer_mission(:NEW.IdMission);
  END IF;
END trg_supprimer_mission;
/
CREATE OR REPLACE PROCEDURE cloturer_convention(p_IdConvention IN NUMBER) AS
BEGIN
  UPDATE Conventions
  SET DateFinConvention = SYSDATE
  WHERE IdConvention = p_IdConvention;
```

```
    DELETE FROM Conventions
    WHERE IdConvention = p_IdConvention;

    COMMIT; -- Valider les modifications
END cloturer_convention;
```