

The cover art features a dark blue background. On the left is a vertical stack of server racks with glowing red indicator lights. In the center, the text 'PROJECT ORACLE DATABASE' is displayed in a light blue, sans-serif font. To the right, there is a glowing orange icon of a database cylinder connected to three smaller boxes, all enclosed in a rounded square frame. A series of flowing, wavy orange lines connect the server racks to the database icon, suggesting data flow. Faint, light blue grid lines are visible in the background.

# PROJECT ORACLE DATABASE

# PROJET

## BASE DE DONNÉES ORACLE

**RÉALISER PAR :**

ACHAIBOU Samy, Costa Alex et Atlan Hugo

# Le contexte

Une entreprise ferroviaire souhaite créer une base de données de gestion centralisée permettant de suivre l'ensemble de son réseau : lignes, trains, wagons, dépôts, employés et stations.

L'objectif de notre base de données est gérer efficacement les informations logistiques, techniques et humaines du réseau ferroviaire.

## Objectifs Globaux

Nous pouvons grâce à notre base de données :

- Gérer (ajouter, modifier, supprimer) toutes les données liées aux entités ferroviaires (lignes, trains, wagons, dépôts, stations, employés).
- Assurer la cohérence entre les différentes entités (exemples : un train appartient à une ligne, un wagon est rattaché à un train, etc.).
- Suivre l'état des lignes et des stations (ouvertes, fermées, en maintenance).
- Connaître les capacités des trains et la disponibilité des wagons.
- Gérer le personnel en les affectant aux différentes lignes, dépôts ou trains.
- Fournir des informations statistiques sur le réseau (nombre de lignes actives, distance totale, nombre de trains opérationnels, etc.).



# Description des besoins

1

## Gestion des lignes

- Chaque ligne est identifiable par un ID\_ligne, et possède un Nom et un État.
- Le chef de ligne doit pouvoir consulter, ajouter, modifier ou désactiver une ou plusieurs lignes
- Une ligne est composée de plusieurs tronçons reliant des stations.

2

## Gestion des trains

- Chaque train est identifiable par un ID\_train, et possède une Vitesse et une Climatisation ou non et un État.
- Chaque train circule sur une ligne et est rattaché à un dépôt.
- Un train est composé au minimum d'un wagon.
- Nous devons savoir la vitesse et savoir s'il est climatisé.

3

## Gestion des wagons

- Chaque wagon est défini par son ID\_Wagon et sont Nombre\_De\_Places et son État.
- Chaque wagon est attaché à un train précis et rattaché à un dépôt.
- Nous pouvons connaître le nombre de places disponibles par wagon et leur état.

4

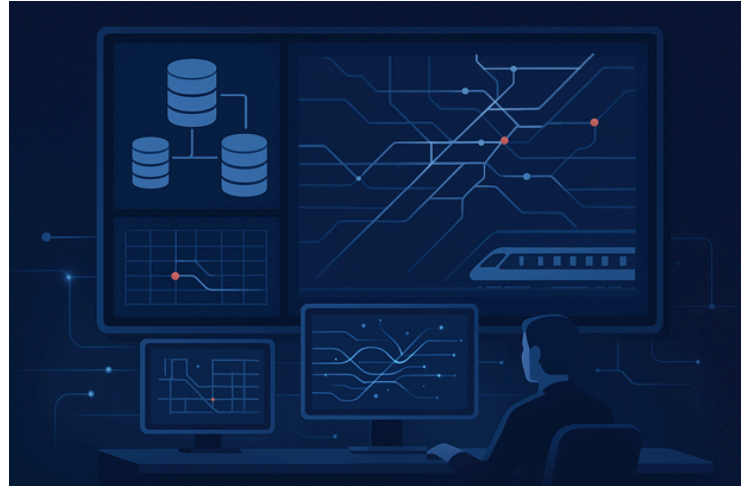
## Gestions des dépôts

- Un dépôt est identifié par son ID\_Dépôt, avec un Nom et une localisation.
- Un dépôt peut accueillir plusieurs trains et wagons.
- La base de données doit permettre d'associer un dépôt à un ou plusieurs employés.

## 5

### Gestion des employés

- Chaque employé est défini par son ID\_Employé, Nom, Prénom, Poste, et Num\_Téléphone.
- Un employé peut être affecté à un dépôt ou à un train selon son poste.
- La base de données doit être capable de suivre les affectations du personnel dans le temps (date de début et de fin).



## 6

### Gestion des stations

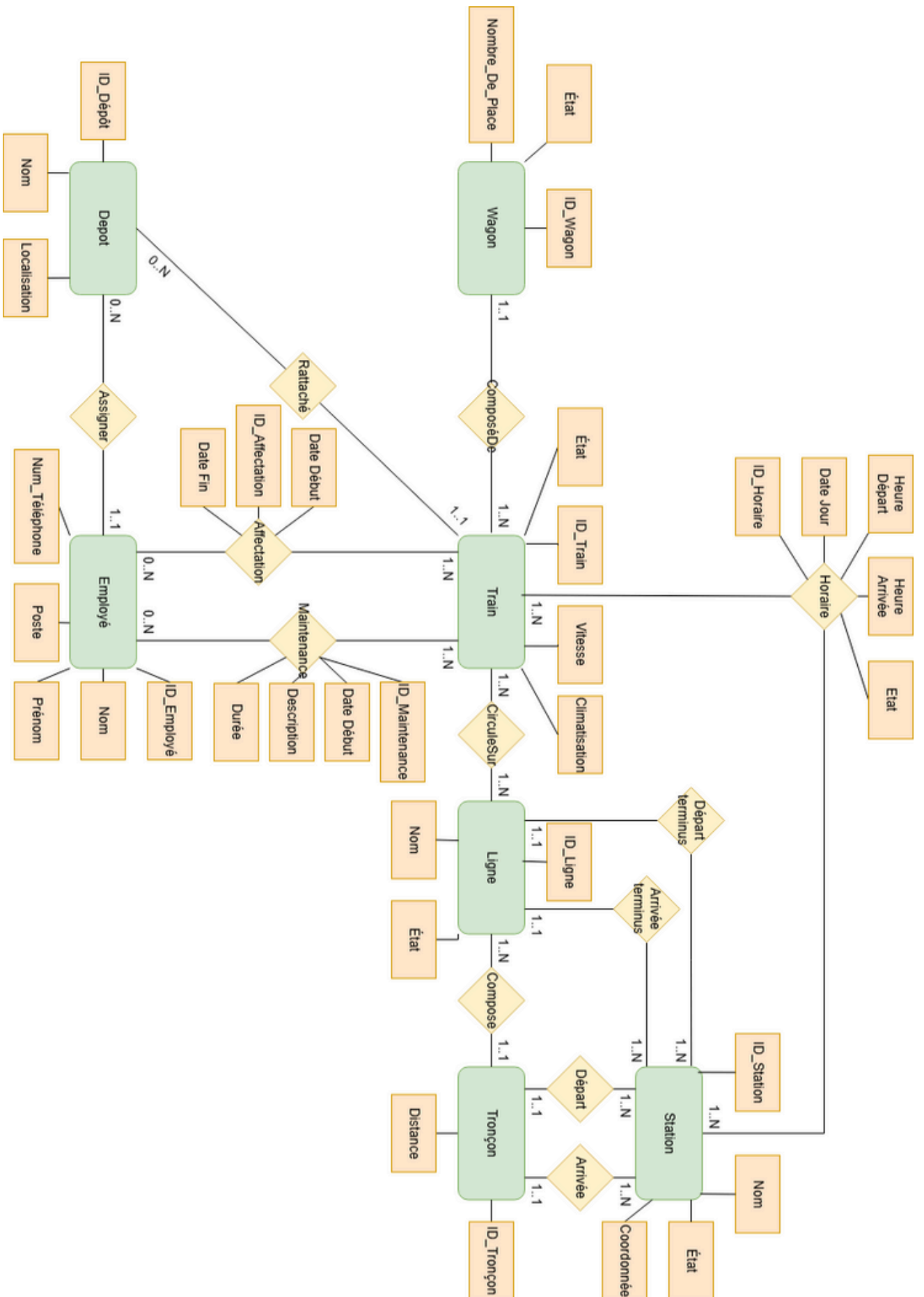
- Chaque station est identifiée par son ID\_Station, avec un Nom, un État et des Coordonnées.
- Les stations appartiennent à une ou plusieurs lignes.
- La base de données doit permettre de suivre leurs états (ouverte, en travaux, fermée) et leur position sur le réseau.

## 7

### Gestion des tronçons

- Un tronçon est défini par son ID\_Tronçon, son Sens et sa Distance.
- Chaque tronçon relie deux stations distinctes : une de départ et une d'arrivée.
- Chaque tronçon appartient à une seule ligne.
- La base de données doit permettre de calculer la distance totale d'une ligne à partir de ses tronçons.

# Schéma Entiter/Association



# Création du schéma de la base de données

1

## Table Dépôt

```
CREATE TABLE Depot (
  ID_Depot NUMBER PRIMARY KEY,
  Nom VARCHAR2(50) NOT NULL,
  Localisation VARCHAR2(100) NOT NULL
);
ALEX
```

2

## Table Station

```
CREATE TABLE Station (
  ID_Station NUMBER PRIMARY KEY,
  Nom VARCHAR2(50) NOT NULL,
  Etat VARCHAR2(10) CHECK (Etat IN ('Ouverte','Fermée','En travaux')) NOT NULL,
  Coordonnee VARCHAR2(30) NOT NULL
);
ALEX
```

3

## Table Ligne

```
CREATE TABLE Ligne (
  ID_Ligne NUMBER PRIMARY KEY,
  Nom VARCHAR2(50) NOT NULL,
  Etat VARCHAR2(15) CHECK (Etat IN ('Fonctionnelle','Interrompue','Perturbée')) NOT NULL,
  Terminus_Depart NUMBER REFERENCES Station(ID_Station) NOT NULL,
  Terminus_Arree NUMBER REFERENCES Station(ID_Station) NOT NULL,
  CONSTRAINT TERMINUS
  CHECK (Terminus_Depart<>Terminus_Arree)
);
ALEX
```

4

## Table Train

```
CREATE TABLE Train (
  ID_Train NUMBER PRIMARY KEY,
  Vitesse NUMBER CHECK (Vitesse >= 0) NOT NULL,
  Climatisation NUMBER CHECK (Climatisation IN (1,0)) NOT NULL,
  Etat VARCHAR2(30) CHECK (Etat IN ('En_Circulation','En_Panne','Au_Depot','En_Reparation')) NOT NULL,
  ID_Ligne NUMBER REFERENCES Ligne(ID_Ligne) NOT NULL,
  ID_Depot NUMBER REFERENCES Depot(ID_Depot) NOT NULL
);
ALEX
```

5

## Table Wagon

```
CREATE TABLE Wagon (
  ID_Wagon NUMBER PRIMARY KEY,
  Nombre_De_Place NUMBER CHECK (Nombre_De_Place>=0) NOT NULL,
  ID_Train NUMBER REFERENCES Train(ID_Train) NOT NULL
);
HUGO
```

6

## Table Employée

```
CREATE TABLE Employe (
  ID_Employe NUMBER PRIMARY KEY,
  Nom VARCHAR2(50) NOT NULL,
  Prenom VARCHAR2(50) NOT NULL,
  Poste VARCHAR2(15) CHECK (Poste IN ('Technicien','Conducteur','Controleur')) NOT NULL,
  Num_Telephone VARCHAR2(15) CHECK (REGEXP_LIKE(Num_Telephone, '^\\+[0-9]{9,14}$')),
  ID_Depot NUMBER REFERENCES Depot(ID_Depot) NOT NULL
);
```

HUGO

7

## Table Tronçon

```
CREATE TABLE Troncon (
  ID_Troncon NUMBER PRIMARY KEY,
  Distance NUMBER CHECK (Distance>0) NOT NULL,
  Station_Depart NUMBER REFERENCES Station(ID_Station) NOT NULL,
  Station_Arrivee NUMBER REFERENCES Station(ID_Station) NOT NULL,
  CONSTRAINT Troncon_Depart_Arrivee
  CHECK (Station_Arrivee<>Station_Depart),
  ID_Ligne NUMBER REFERENCES Ligne(ID_Ligne) NOT NULL
);
```

HUGO

8

## Table Horaire

```
CREATE TABLE Horaire (
  ID_Horaire NUMBER PRIMARY KEY,
  Heure_Arrivee TIMESTAMP NOT NULL,
  Heure_Depart TIMESTAMP NOT NULL,
  Date_Horaire DATE NOT NULL,
  CONSTRAINT Horaire_Arr_Dep
  CHECK (Heure_Arrivee>Heure_Depart),
  Etat VARCHAR2(30) CHECK (Etat IN ('En_Retard','Supprimé','Présent')) NOT NULL,
  ID_Train NUMBER REFERENCES Train(ID_Train) NOT NULL,
  ID_Station NUMBER REFERENCES Station(ID_Station) NOT NULL
);
```

SAMY

9

## Table Affectation

```
CREATE TABLE Affectation (
  ID_Affectation NUMBER PRIMARY KEY,
  Date_Debut DATE NOT NULL,
  Date_Fin DATE NOT NULL,
  ID_Train NUMBER REFERENCES Train(ID_Train),
  ID_Employe NUMBER REFERENCES Employe(ID_Employe)
);
```

SAMY

10

## Table Maintenance

```
CREATE TABLE Maintenance (
  ID_Maintenance NUMBER PRIMARY KEY,
  Date_Maint DATE NOT NULL,
  Description CLOB,
  Duree NUMBER NOT NULL,
  ID_Train NUMBER REFERENCES Train(ID_Train) NOT NULL,
  ID_Employe NUMBER REFERENCES Employe(ID_Employe) NOT NULL
);
```

SAMY

## 1

### Composition des données

Le jeu de données contient au minimum 30 tuples par table afin de tester les requêtes complexes.

voici le lien GitHub pour voir intégralités des données :

<https://github.com/Hugoat22/Donn-e-pour-projet-oracle>

HUGO

## 2

### Insertion SQL

```
INSERT INTO Ligne VALUES (17,B,Fonctionnel,Mitry - Claye,Saint-Rémy-lès-Chevreuse);
```

ALEX

```
INSERT INTO Station VALUES (5,Massy - Verrières,Ouverte,"2.27405499755523,48.73485498137642");
```

HUGO

```
INSERT INTO Employe VALUES (49,Moreau,Antoine,Conducteur,+737568824,2);
```

SAMY

## 3

### SQL LOAD

```
LOAD DATA
INFILE 'lignes.csv'
APPEND
INTO TABLE Ligne
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
(ID_Ligne, Nom, Etat, Terminus_Depart, Terminus_Arrivee)
```

ALEX

```
LOAD DATA
INFILE 'stations.csv'
APPEND
INTO TABLE Station
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
(ID_Station, Nom, Etat, Coordonnee)
```

HUGO

```
LOAD DATA
INFILE 'employées.csv'
APPEND
INTO TABLE Employe
FIELDS TERMINATED BY ',' OPTIONALLY ENCLOSED BY '"'
(ID_Employe, Nom, Prenom, Poste, Num_Telephone, ID_Depot)
```

SAMY

# Contraintes d'intégrité

1

## Cohérence des places

**Un wagon doit avoir au moins un place.**

```
CREATE OR REPLACE TRIGGER wagon_places
BEFORE INSERT OR UPDATE ON Wagon FOR EACH ROW
BEGIN
    IF :NEW.Nombre_De_Place <= 0 THEN
        RAISE_APPLICATION_ERROR(-20001,'Un wagon doit avoir au moins
une place !');
    END IF;
END;
/
```

ALEX

2

## Cohérence de maintenance

**Un train ne peut circuler que s'il n'est pas en maintenance actuellement.**

```
CREATE OR REPLACE TRIGGER Maintenance_Train
BEFORE INSERT OR UPDATE ON Train FOR EACH ROW
DECLARE
    nb_train_maint NUMBER;
BEGIN
    SELECT COUNT(*) INTO nb_train_maint
    FROM Maintenance M
    WHERE M.ID_Train = :NEW.ID_Train AND SYSDATE BETWEEN Date_Maint AND
(Date_Maint + Duree);
    IF nb_train_maint > 0 THEN
        RAISE_APPLICATION_ERROR(-20002,'Impossible de mettre le train en circulation :
maintenance en cours !');
    END IF;
END;
/
```

ALEX

3

## Validité du nombres de wagons

**Un train peut avoir maximum que 3 wagon attacher a lui.**

```
CREATE OR REPLACE TRIGGER Train_Max_Wagon
BEFORE INSERT OR UPDATE ON Wagon FOR EACH ROW
DECLARE
    nb_wagons NUMBER;
BEGIN
    SELECT COUNT(*) INTO nb_wagons
    FROM Wagon W
    WHERE W.ID_Train = :NEW.ID_Train AND W.ID_Wagon != :NEW.ID_Wagon;
    IF nb_wagons >= 3 THEN
        RAISE_APPLICATION_ERROR(-20003,'La limite de 3 wagons pour ce train est atteinte
!');
    END IF;
END;
/
```

ALEX

## 4

## Validité du nombres des dépôts

### Un dépôt a une capacité maximum de 10 train et 25 employer maximum .

```
CREATE OR REPLACE TRIGGER Depot_Max_Train
BEFORE INSERT OR UPDATE ON Train FOR EACH ROW
DECLARE
    nb_train NUMBER;
BEGIN
    SELECT COUNT(*) INTO nb_train
    FROM Train T
    WHERE T.ID_Depot = :NEW.ID_Depot AND T.ID_Train != :NEW.ID_Train;
    IF nb_train >= 10 THEN
        RAISE_APPLICATION_ERROR(-20004,'La limite de 10 Train pour ce dépôt est atteinte !');
    END IF;
END;
/

CREATE OR REPLACE TRIGGER Depot_Max_Employe
BEFORE INSERT OR UPDATE ON Employe FOR EACH ROW
DECLARE
    nb_employe NUMBER;
BEGIN
    SELECT COUNT(*) INTO nb_employe
    FROM Employe E
    WHERE E.ID_Depot = :NEW.ID_Depot AND E.ID_Employe != :NEW.ID_Employe;
    IF nb_employe >= 25 THEN
        RAISE_APPLICATION_ERROR(-20014,'La limite de 25 Employée pour ce dépôt est atteinte !');
    END IF;
END;
/
```

HUGO

## 5

## Disponibilité des employés

### Un employé ne peut pas être affecté à deux trains différents sur la même période.

```
CREATE OR REPLACE TRIGGER Distinctes_Travaille_Employe
BEFORE INSERT OR UPDATE ON Affectation FOR EACH ROW
DECLARE
    nb_affect NUMBER;
BEGIN
    SELECT COUNT(*) INTO nb_affect
    FROM Affectation
    WHERE ID_Employe = :NEW.ID_Employe AND ID_Train != :NEW.ID_Train
    AND (:NEW.Date_Debut <= Date_Fin AND :NEW.Date_Fin >= Date_Debut)
    AND ID_Affectation != :NEW.ID_Affectation;

    IF nb_affect > 0 THEN
        RAISE_APPLICATION_ERROR(-20005,'Employé déjà affecté à un autre train sur cette période !');
    END IF;
END;
/
```

HUGO

## 6

## Validité horaire

### Pour un horaire donné, l'heure d'arrivée doit être supérieur à l'heure de départ.

```
CREATE OR REPLACE TRIGGER Horaire_Durée_Positif
BEFORE INSERT OR UPDATE ON Horaire FOR EACH ROW
BEGIN
    IF :NEW.Heure_Depart >= :NEW.Heure_Arrivee THEN
        RAISE_APPLICATION_ERROR(-20006,'L"heure de départ doit être strictement inférieur à l"heure d"arrivée !');
    END IF;
END;
/
```

HUGO

## 7

## Structure des lignes

**Chaque ligne doit avoir des tronçons qui ont comme station de départ la station d'arrivée du tronçon précédent**

```
CREATE OR REPLACE TRIGGER Troncon_Connecte
BEFORE INSERT OR UPDATE ON Troncon FOR EACH ROW
DECLARE
    terminus_depart NUMBER;
    nb_troncon NUMBER;
BEGIN
    SELECT Terminus_Depart INTO terminus_depart
    FROM Ligne
    WHERE ID_Ligne = :NEW.ID_Ligne;

    IF :NEW.Station_Depart != terminus_depart THEN
        SELECT COUNT(*) INTO nb_troncon
        FROM Troncon
        WHERE ID_Ligne = :NEW.ID_Ligne AND Station_Arrivee = :NEW.Station_Depart;

        IF nb_troncon = 0 THEN
            RAISE_APPLICATION_ERROR(-20007,'La station de départ doit correspondre à la station d'arrivée
            d'un tronçon précédent');
        END IF;
    END IF;
END;
/
```

SAMY

## 8

## Cohérence des terminus

**Un terminus ne peut pas être à la fois départ et arrivée de la même ligne.**

```
CREATE OR REPLACE TRIGGER Distinctes_Terminus
BEFORE DELETE OR UPDATE ON Ligne
FOR EACH ROW
BEGIN
    IF :NEW.Terminus_Depart = :NEW.Terminus_Arrivee THEN
        RAISE_APPLICATION_ERROR(-2008,'Le terminus de départ et d'arrivée doit être différent !');
    END IF;
END;
/
```

SAMY

## 9

## Cohérence du travail

**Un employé selon son poste n'aura pas la même relation avec le train.**

```
CREATE OR REPLACE TRIGGER Poste_Maintenance
BEFORE INSERT OR UPDATE ON Maintenance
FOR EACH ROW
BEGIN
    IF EXISTS(
        SELECT 1
        FROM Employe E
        WHERE E.ID_Employe = :NEW.ID_Employe AND E.Poste != 'Technicien'
    ) THEN
        RAISE_APPLICATION_ERROR(-20009,'Cet employé ne peut pas travailler à la maintenance !');
    END IF;
END;
/
```

```
CREATE OR REPLACE TRIGGER Poste_Affectation
BEFORE INSERT OR UPDATE ON Affectation
FOR EACH ROW
BEGIN
    IF EXISTS(
        SELECT 1
        FROM Employe E
        WHERE E.ID_Employe = :NEW.ID_Employe AND E.Poste = 'Technicien'
    ) THEN
        RAISE_APPLICATION_ERROR(-20019,'Cet employé ne peut travailler qu'à la maintenance !');
    END IF;
END;
/
```

SAMY

# Droits d'utilisateurs

UTILISATEUR	Rôles	Droits
<b>Administrateur</b>	Supervise le réseau complet	<b>Accès à toute les vues</b>
<b>Chef de dépôt</b>	Supervise les trains, wagons et maintenance	<b>Accès à Vue_Trains_Dépôt, Vue_Maintenance_Récente et Vue_Affectation</b>
<b>Chef de ligne</b>	Gère les lignes, stations et horaires	<b>Accès à Vue_Lignes_Actives, Vue_Horaires_Ligne et Vue_Trains_Sur_Ligne</b>
<b>Technicien</b>	Réalise la maintenance du matériel	<b>Accès à Vue_Maintenance_Récente</b>
<b>Conducteur/Contrôleur</b>	Consulte ses affectations et horaires	<b>Accès à Vue_Affectation et Vue_Horaires_Ligne</b>

```

CREATE USER Administrateur IDENTIFIED BY Admini;
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP
QUOTA UNLIMITED ON USERS;
CREATE USER Chef_De_Depot IDENTIFIED BY Chef_Depot;
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP
QUOTA UNLIMITED ON USERS;
GRANT CREATE SESSION TO Administrateur;
GRANT CREATE SESSION TO Chef_De_Depot;
CREATE ROLE Role_Admin;
CREATE ROLE Role_chef_Depot;
GRANT Role_Admin TO Administrateur;
GRANT Role_chef_Depot TO Chef_De_Depot;
ALEX
CREATE USER Chef_De_Ligne IDENTIFIED BY Chef_Ligne;
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP
QUOTA UNLIMITED ON USERS;
CREATE USER Technicien IDENTIFIED BY Tech;
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP
QUOTA UNLIMITED ON USERS;
GRANT CREATE SESSION TO Chef_De_Ligne;
GRANT CREATE SESSION TO Technicien;
CREATE ROLE Role_chef_Ligne;
CREATE ROLE Role_Technicien;
GRANT Role_chef_Ligne TO Chef_De_Ligne;
GRANT Role_Technicien TO Technicien;
HUGO
CREATE USER Conduc_Control IDENTIFIED BY cond_cont;
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE TEMP
QUOTA UNLIMITED ON USERS;
GRANT CREATE SESSION TO Conduc_Control;
CREATE ROLE Role_Cond_Cont;
GRANT Role_Cond_Cont TO Conduc_Control;
SAMMY

```

# Exemple de Vues

1

## Vue\_Lignes\_Actives

**Cette vue permet de voir facilement les lignes actives en affichant les lignes dont l'état est "Fonctionnelle" ainsi que le nom des stations de départ et d'arrivée.**

```
CREATE OR REPLACE VIEW VUE_LIGNES_ACTIVES AS
SELECT
  L.ID_Ligne,
  L.Nom,
  L.Etat,
  SD.Nom AS Station_Depart,
  SA.Nom AS Station_Arrivee
FROM Ligne L, Station SD, Station SA
WHERE L.Terminus_Depart= SD.ID_Station AND L.Terminus_Arrivee= SA.ID_Station AND
L.Etat = 'Fonctionnelle';
```

```
GRANT SELECT ON VUE_LIGNES_ACTIVES TO Role_Admin;
GRANT SELECT ON VUE_LIGNES_ACTIVES TO Role_chef_Ligne;
ALEX
```

2

## Vue\_Trains\_Dépôt

**Cette vue permet de voir à quel dépôt appartient chaque train en affichant la liste des trains et le nom du dépôt auquel ils sont rattachés.**

```
CREATE OR REPLACE VIEW VUE_Trains_Depot AS
SELECT
  T.ID_Train,
  D.Nom AS Nom_Depot
FROM Train T, Depot D
WHERE T.ID_Depot=D.ID_Depot;
```

```
GRANT SELECT ON VUE_Trains_Depot TO Role_Admin;
GRANT SELECT ON VUE_Trains_Depot TO Role_chef_Depot;
ALEX
```

3

## Vue\_Maintenance\_Récente

**Cette vue va permettre de connaître les opérations de maintenance réalisées au cours du dernier mois en affichant toutes les maintenances qui concernent un train.**

```
CREATE OR REPLACE VIEW Vue_Maintenance_Recente AS
SELECT
  M.ID_Maintenance,
  M.Date_Maint,
  M.ID_Train
FROM Maintenance M
WHERE M.Date_Maint>=ADD_MONTHS(TRUNC(SYSDATE, 'MM'),-1)
AND M.Date_Maint<TRUNC(SYSDATE,'MM');
```

```
GRANT SELECT ON Vue_Maintenance_Recente TO Role_Admin;
GRANT SELECT ON Vue_Maintenance_Recente TO Role_chef_Depot;
GRANT SELECT ON Vue_Maintenance_Recente TO Role_Technicien;
HUGO
```

## 4

## Vue\_Affectation

**Cette vue permet de voir les trains sur lesquels les employés sont affectés en affichant la liste des trains avec ses principales informations et l'identité des employés**

```
CREATE OR REPLACE VIEW Vue_Affectation AS
SELECT
  E.ID_Employe,
  E.Nom || ' ' || E.Prenom AS Nom_Complet,
  T.ID_Train,
  T.Vitesse,
  T.Climatisation,
  T.Etat AS Etat_Train,
  T.ID_Ligne
FROM Employe E, Affectation A, Train T
WHERE E.ID_Employe = A.ID_Employe AND A.ID_Train = T.ID_Train;
```

```
GRANT SELECT ON Vue_Affectation TO Role_Admin;
GRANT SELECT ON Vue_Affectation TO Role_chef_Depot;
GRANT SELECT ON Vue_Affectation TO Role_Cond_Cont;
HUGO
```

## 5

## Vue\_Horaires\_Ligne

**Cette vue permet de voir les horaires par ligne en affichant toutes les informations d'horaires pour chaque station et ligne**

```
CREATE OR REPLACE VIEW Vue_Horaires_Ligne AS
SELECT
  L.ID_Ligne,
  L.Nom AS Nom_Ligne,
  T.ID_Train,
  S.Nom AS Nom_Station,
  H.Heure_Depart,
  H.Heure_Arrivee,
  H.Date_Horaire,
  H.Etat AS Etat_Horaire
FROM Ligne L, Train T, Horaire H, Station S
WHERE T.ID_Ligne = L.ID_Ligne AND H.ID_Train = T.ID_Train AND S.ID_Station =
H.ID_Station;
```

```
GRANT SELECT ON Vue_Horaires_Ligne TO Role_Admin;
GRANT SELECT ON Vue_Horaires_Ligne TO Role_chef_Ligne;
GRANT SELECT ON Vue_Horaires_Ligne TO Role_Cond_Cont;
SAM Y
```

## 6

## Vue\_Trains\_Sur\_Ligne

**Cette vue permet de voir les trains en circulation sur une ligne en affichant toutes les informations sur le train**

```
CREATE OR REPLACE VIEW Vue_Trains_Sur_Ligne AS
SELECT
  ID_Ligne
  ID_Train,
  Vitesse,
  Climatisation,
  Etat,
  ID_Depot
FROM Train
WHERE Etat = 'En_Circulation'
ORDER BY ID_Train;
```

```
GRANT SELECT ON Vue_Trains_Sur_Ligne TO Role_Admin;
GRANT SELECT ON Vue_Trains_Sur_Ligne TO Role_chef_Ligne;
SAM Y
```

# Requêtes attendues

**Rechercher le nombre total de trains rattachés à chaque dépôt, ainsi que leur vitesse moyenne.**

```
SELECT d.ID_Depot, d.Nom,
       COUNT(t.ID_Train) AS nb_trains,
       AVG(t.Vitesse) AS vitesse_moyenne
FROM Depot d
LEFT JOIN Train t ON d.ID_Depot = t.ID_Depot
GROUP BY d.ID_Depot, d.Nom
ALEX
```

---

**identifier le nombre d'employés affectés à chaque dépôt.**

```
SELECT d.ID_Depot, d.Nom,
       COUNT(DISTINCT e.ID_Employe) AS nb_employes
FROM Depot d
LEFT JOIN Employe e ON d.ID_Depot = e.ID_Depot
GROUP BY d.ID_Depot, d.Nom;
ALEX
```

---

**Trouver les trains ayant subi plusieurs opérations de maintenance et afficher le nombre de maintenances par train.**

```
SELECT t.ID_Train,
       COUNT(m.ID_Maintenance) AS nb_maintenances
FROM Train t, Maintenance m
WHERE t.ID_Train = m.ID_Train
GROUP BY t.ID_Train
HAVING COUNT(m.ID_Maintenance) > 1;
ALEX
```

---

**Lister les lignes sur lesquelles circulent plusieurs trains et afficher la capacité totale moyenne des trains de chaque ligne.**

```
SELECT l.ID_Ligne, l.Nom,
       COUNT(DISTINCT t.ID_Train) AS nb_trains,
       AVG(w.Nombre_De_Place) AS capacite_moyenne
FROM Ligne l, Train t, Wagon w
WHERE l.ID_Ligne = t.ID_Ligne AND t.ID_Train = w.ID_Train
GROUP BY l.ID_Ligne, l.Nom
HAVING COUNT(t.ID_Train) > 1;
ALEX
```

---

**Identifier les trains qui n'ont pas eu de maintenance depuis plus de 6 mois.**

```
SELECT t.ID_Train
FROM Train t
WHERE NOT EXISTS (
  SELECT 1
  FROM Maintenance m
  WHERE m.ID_Train = t.ID_Train
  AND m.Date_Maint >= ADD_MONTHS(SYSDATE, -6)
);
ALEX
```

---

**Donner le nombre de wagons composant chaque train et la moyenne du nombre de places par wagon.**

```
SELECT t.ID_Train,
       COUNT(w.ID_Wagon) AS nb_wagons,
       AVG(w.Nombre_De_Place) AS moyenne_places
FROM Train t
LEFT JOIN Wagon w ON t.ID_Train = w.ID_Train
GROUP BY t.ID_Train;
ALEX
```

---

**Rechercher l'employés ayant été affectés au plus grand nombre affectation dans une journée donné.**

```
SELECT *
FROM (
  SELECT a.ID_Employe, COUNT(*) AS nb_affectations
  FROM Affectation a
  WHERE a.Date_Debut = DATE '2025-12-15'
  GROUP BY a.ID_Employe
  ORDER BY nb_affectations DESC
)
WHERE ROWNUM = 1;
HUGO
```

---

**Trouver les trains dont la capacité totale est supérieure à la moyenne de toutes les capacités**

```
SELECT t.ID_Train, SUM(w.Nombre_De_Place) AS capacite_totale
FROM Train t, Wagon w
WHERE t.ID_Train = w.ID_Train
GROUP BY t.ID_Train
HAVING SUM(w.Nombre_De_Place) >
  (SELECT AVG(cap)
   FROM (
     SELECT SUM(Nombre_De_Place) AS cap
     FROM Wagon
     GROUP BY ID_Train
   )
   sub
  );
HUGO
```

---

**Lister chaque ligne et le nombre de stations (départ et arrivée confondus) qu'elle dessert.**

```
SELECT l.ID_Ligne, l.Nom,
       COUNT(DISTINCT s.ID_Station) AS nb_stations
FROM Ligne l, Troncon t, Station s
WHERE l.ID_Ligne = t.ID_Ligne AND s.ID_Station IN (t.Station_Depart, t.Station_Arrivee)
GROUP BY l.ID_Ligne, l.Nom;
HUGO
```

---

**Identifier les station utilisés par plusieurs lignes et afficher la liste des lignes correspondantes.**

```
SELECT s.ID_Station, s.Nom, COUNT(DISTINCT t.ID_Ligne) AS nb_lignes
FROM Station s, Troncon t
WHERE s.ID_Station IN (t.Station_Depart, t.Station_Arrivee)
GROUP BY s.ID_Station, s.Nom
HAVING COUNT(DISTINCT t.ID_Ligne) > 1;
HUGO
```

---

**Trouver les lignes actuellement en service (État = 'Fonctionnelle') et la distance totale parcourue (somme des distances des tronçons).**

```
SELECT l.ID_Ligne, l.Nom,
       SUM(t.Distance) AS distance_totale
FROM Ligne l, Troncon t
WHERE l.ID_Ligne = t.ID_Ligne AND l.Etat = 'Fonctionnelle'
GROUP BY l.ID_Ligne, l.Nom;
HUGO
```

---

**Afficher la liste des employés ayant effectué des maintenances et le nombre total d'interventions par employé.**

```
SELECT e.ID_Employe, e.Nom,
       COUNT(m.ID_Maintenance) AS total_interventions
FROM Employe e, Maintenance m
WHERE e.ID_Employe = m.ID_Employe
GROUP BY e.ID_Employe, e.Nom;
HUGO
```

---

**Identifier les trains en circulation dans un intervalle de temps donné.**

```
SELECT DISTINCT h.ID_Train
FROM Horaire h
WHERE TO_CHAR(h.Heure_Depart, 'HH24:MI') BETWEEN '08:00' AND '12:00';
SAMMY
```

---

**Donner les 5 employés ayant participé au plus grand nombre d'affectations dans l'ordre croissant et afficher leur total.**

```
SELECT *
FROM (
  SELECT e.ID_Employe, e.Nom,
         COUNT(a.ID_Affectation) AS total
  FROM Employe e, Affectation a
  WHERE e.ID_Employe = a.ID_Employe
  GROUP BY e.ID_Employe, e.Nom
  ORDER BY total ASC
)
WHERE ROWNUM = 5;
SAMMY
```

---

**Lister les lignes qui n'ont actuellement aucun train circulant dessus.**

```
SELECT l.ID_Ligne, l.Nom
FROM Ligne l
LEFT JOIN Train t ON l.ID_Ligne = t.ID_Ligne
WHERE t.ID_Train IS NULL;
SAMMY
```

---

**Calculer la durée moyenne, minimale et maximale des maintenances pour chaque ligne.**

```
SELECT l.ID_Ligne, l.Nom,
       AVG(m.Duree) AS duree_moyenne,
       MIN(m.Duree) AS duree_min,
       MAX(m.Duree) AS duree_max
FROM Ligne l, Train t, Maintenance m
WHERE l.ID_Ligne = t.ID_Ligne AND t.ID_Train = m.ID_Train
GROUP BY l.ID_Ligne, l.Nom;
SAMY
```

---

**Trouver les wagons qui ne sont associés à aucun train.**

```
SELECT ID_Wagon
FROM Wagon
WHERE ID_Train IS NULL;
SAMY
```

---

**Afficher la liste des postes distincts occupés par les employés dans l'entreprise ainsi que le nombre d'employé par poste.**

```
SELECT Poste,
       COUNT(ID_Employe) AS nb_employes
FROM Employe
GROUP BY Poste;
SAMY
```

---

**Classer les trains selon leur vitesse en catégories : "Rapide"(>90), "Moyen"(>85) ou "Lent".**

```
SELECT ID_Train,
       CASE
         WHEN Vitesse > 90 THEN 'Rapide'
         WHEN Vitesse > 85 THEN 'Moyen'
         ELSE 'Lent'
       END AS categorie
FROM Train;
SAMY
```

---

**Trouver les dépôts sans aucun train rattaché.**

```
SELECT d.ID_Depot, d.Nom
FROM Depot d
LEFT JOIN Train t ON d.ID_Depot = t.ID_Depot
WHERE t.ID_Train IS NULL;
SAMY
```

---

# META / DONNÉES

**1/script SQL qui donne une fois exécuté la liste de toutes les contraintes d'intégrité définies sur votre BD que vous classerez par table et par type de contrainte, le corps de chaque contrainte devra être récupéré.**

```
SELECT
    c.table_name,
    c.constraint_name,
    c.constraint_type,
    c.search_condition
FROM user_constraints c
ORDER BY c.table_name, c.constraint_type;
```

ALEX

**2/script SQL qui donne une fois exécuté la liste de tous les triggers que vous avez définis, classés par nom de table.**

```
SELECT
    trigger_name,
    table_name,
    trigger_type,
    triggering_event,
    status,
    description,
    trigger_body
FROM user_triggers
ORDER BY table_name, trigger_name;
```

HUGO

**3/script SQL qui donne une fois exécuté la liste de tous les triggers que vous avez définis, classés par nom de table.**

```
SELECT
    table_name,
    num_rows
FROM user_tables
ORDER BY table_name;
```

SAMY

# Traduction du schéma E/A en modèle relationnel

Ligne (**ID\_Ligne**, Nom, État, Terminus\_Départ, Terminus\_Arrivée)  
 Train (**ID\_Train**, Vitesse, Climatisation, État, ID\_Ligne, ID\_Dépôt)  
 Wagon (**ID\_Wagon**, Nombre\_De\_Place, État, ID\_train)  
 Dépôt (**ID\_Dépôt**, Nom, Localisation)  
 Employé (**ID\_Employé**, Nom, Prénom, Poste, Num\_Téléphone, ID\_Dépôt)  
 Station (**ID\_Station**, Nom, État, Coordonnée)  
 Tronçon (**ID\_Tronçon**, Distance, Station\_Départ, Station\_Arrivée, ID\_Ligne)  
 Horaire (**ID\_Horaire**, Heure\_Arrivée, Heure\_Départ, Date\_Jour, ID\_Train, ID\_Station)  
 Affectation (**ID\_Affectation**, Date\_Début, Date\_Fin, ID\_Train, ID\_Employé)  
 Maintenance (**ID\_Maintenance**, Date\_Début, Date\_Fin, Description, ID\_Train, ID\_Employé)

Ligne (Terminus\_Départ) est une clé étrangère de Station (ID\_Station)  
 Ligne (Terminus\_Arrivée) est une clé étrangère de Station (ID\_Station)  
 Train (ID\_Ligne) est une clé étrangère de Ligne (ID\_Ligne)  
 Train (ID\_Dépôt) est une clé étrangère de Dépôt (ID\_Dépôt)  
 Wagon (ID\_Train) est une clé étrangère de Train (ID\_Train)  
 Wagon (ID\_Dépôt) est une clé étrangère de Dépôt (ID\_Dépôt)  
 Employé (ID\_Dépôt) est une clé étrangère de Dépôt (ID\_Dépôt)  
 Tronçon (Station\_Départ) est une clé étrangère de Station (ID\_Station)  
 Tronçon (Station\_Arrivée) est une clé étrangère de Station (ID\_Station)  
 Tronçon (ID\_Ligne) est une clé étrangère de Ligne (ID\_Ligne)  
 Horaire (ID\_Train) est une clé étrangère de Train (ID\_Train)  
 Horaire (ID\_Station) est une clé étrangère de Station (ID\_Station)  
 Affectation (ID\_Train) est une clé étrangère de Train (ID\_Train)  
 Affectation (ID\_Employé) est une clé étrangère de Employé (ID\_Employé)  
 Maintenance (ID\_Train) est une clé étrangère de Train (ID\_Train)  
 Maintenance (ID\_Employé) est une clé étrangère de Employé (ID\_Employé)