RAPPORT DE BASE DE DONNÉES

PROJET S301

Réalisé par Nesrine CHARLES, Selman BOUZLAFA et Rached DAHMANI.

Groupe 3D

SOMMAIRE

Introduction

À chaque rentrée universitaire, les responsables pédagogiques doivent constituer manuellement les groupes de TD et TP à partir des étudiants inscrits. Cette tâche, complexe et chronophage, doit respecter de multiples contraintes : homogénéité du niveau, équilibre des genres, répartition des baccalauréats, covoiturage, etc.

Le projet S301 a pour objectif de concevoir une plateforme numérique permettant d'automatiser et faciliter cette création de groupes, tout en offrant une interface simple pour les étudiants et les enseignants.

On cherche, par ce projet, à proposer des interfaces web et une base de données résistante.

Dans le cadre du projet S301 du BUT 2 Informatique, nous devons établir plusieurs documents présentant les différentes phases du projet.

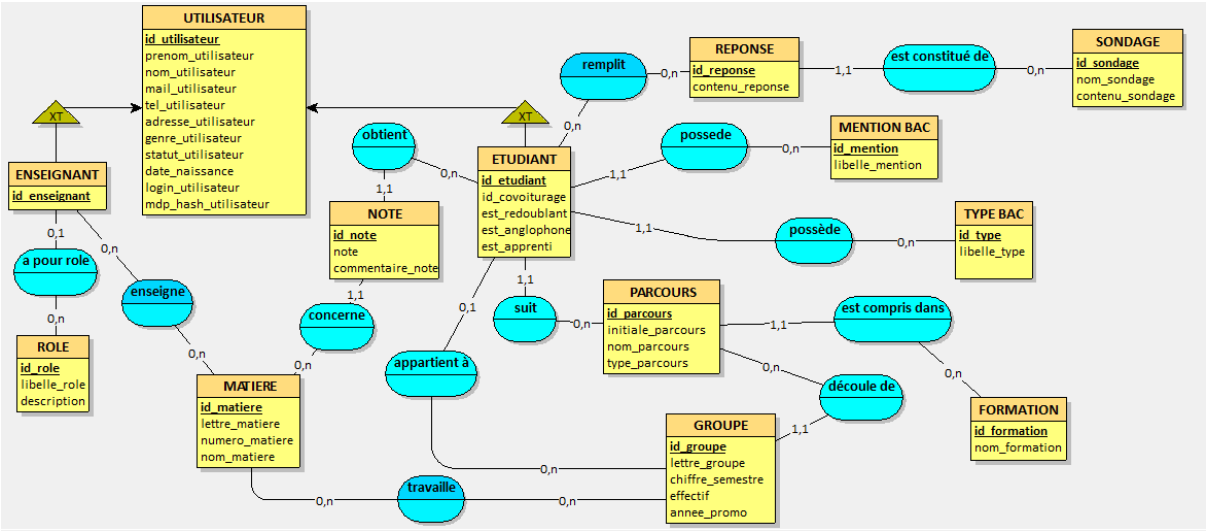
Ce deuxième document concerne la partie "**Base de Données**". Cette partie à pour but d'implémenter une base de données et vérifier que cette implémentation répond aux différentes fonctionnalités. La vérification des fonctionnalités se fait à partir de requêtes sous-jacentes écrites par notre équipe, ce qui permet éventuellement de corriger les manquements.

Partie I : Implémentation de la base de données, aspect statique

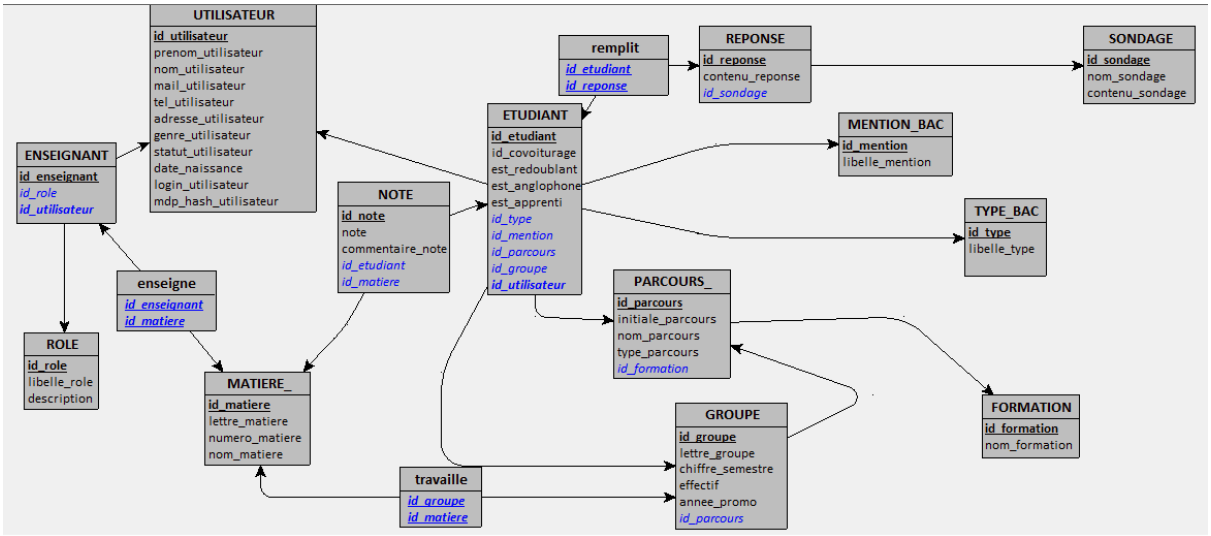
Schéma Relationnel (SR) commenté

Avant de commencer le schéma relationnel (SR), nous avons dû apporter quelques modifications concernant le MCD. Le changement le plus urgent concerne les sondages : on a corrigé les relations pour permettre aux étudiants de répondre à plusieurs questionnaires, ce qui n'était pas possible avant. On a aussi simplifié la gestion des enseignants en remplaçant l'héritage, trop rigide, par un système de rôles qui permet de cumuler les responsabilités facilement. Enfin, on a profité de cette refonte pour intégrer les données techniques manquantes, comme la sécurité des mots de passe et les critères nécessaires à la répartition des groupes.

Voici le nouveau MCD :



Nous avons réalisé le schéma relationnel, ensuite, à partir de notre MCD et ses modifications, sur Looping :



L'application Looping intègre une création de script à partir d'un MCD pour créer les tables dans la base de données. Nous avons donné notre MCD afin de réaliser le script suivant :

```
-- NETTOYAGE PREALABLE
BEGIN
  EXECUTE IMMEDIATE 'DROP TABLE MATIERE_GROUPE CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE ENSEIGNANT_MATIERE CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE ETUDIANT_REPONSE CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE NOTE CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE ETUDIANT CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE GROUPE CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE ENSEIGNANT CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE TYPE_BAC CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE ROLE CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE MENTION_BAC CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE REPONSE CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE MATIERE CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE PARCOURS CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE SONDAGE CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE FORMATION CASCADE CONSTRAINTS';
  EXECUTE IMMEDIATE 'DROP TABLE UTILISATEUR CASCADE CONSTRAINTS';
EXCEPTION
  WHEN OTHERS THEN NULL; -- Ignore les erreurs si les tables n'existent pas encore
END;
```

```
/

-- UTILISATEUR
CREATE TABLE UTILISATEUR(
   id_utilisateur NUMBER(10),
   prenom_utilisateur VARCHAR2(50) NOT NULL,
   nom_utilisateur VARCHAR2(50) NOT NULL,
   mail_utilisateur VARCHAR2(100) NOT NULL,
   tel_utilisateur VARCHAR2(15),
   adresse_utilisateur VARCHAR2(200),
   genre_utilisateur VARCHAR2(20),
   statut_utilisateur VARCHAR2(20),        -- Exemple: 'ETUDIANT', 'ENSEIGNANT'
   date_naissance DATE,
   login_utilisateur VARCHAR2(50) NOT NULL,
   mdp_hash_utilisateur VARCHAR2(255) NOT NULL,
   PRIMARY KEY(id_utilisateur)
);

-- FORMATION
CREATE TABLE FORMATION(
   id_formation VARCHAR2(10),
   nom_formation VARCHAR2(100), -- Exemple: 'Informatique'
   PRIMARY KEY(id_formation)
);

-- SONDAGE
CREATE TABLE SONDAGE(
   id_sondage NUMBER(10),
   nom_sondage VARCHAR2(100),
   contenu_sondage VARCHAR2(4000),
   PRIMARY KEY(id_sondage)
);

-- PARCOURS
CREATE TABLE PARCOURS(
   id_parcours VARCHAR2(10),
   initiale_parcours VARCHAR2(5), -- Exemple: 'A'
   nom_parcours VARCHAR2(100),
   type_parcours VARCHAR2(10), -- Exemple: 'Initial' ou 'Apprenti'
   id_formation VARCHAR2(10) NOT NULL,
   PRIMARY KEY(id_parcours),
   FOREIGN KEY(id_formation) REFERENCES FORMATION(id_formation)
```

```
);

-- MATIERE
CREATE TABLE MATIERE(
   id_matiere VARCHAR2(20), -- Exemple: 'R3.07'
   lettre_matiere CHAR(1),  -- Exemple: 'A'
   numero_matiere VARCHAR2(10),
   nom_matiere VARCHAR2(100),
   PRIMARY KEY(id_matiere)
);

-- REPONSE
CREATE TABLE REPONSE(
   id_reponse NUMBER(10),
   contenu_reponse VARCHAR2(500), -- Réponses parfois longues
   id_sondage NUMBER(10) NOT NULL,
   PRIMARY KEY(id_reponse),
   FOREIGN KEY(id_sondage) REFERENCES SONDAGE(id_sondage)
);

-- MENTION_BAC
CREATE TABLE MENTION_BAC(
   id_mention VARCHAR2(10),
   libelle_mention VARCHAR2(50), -- Exemple: 'Très Bien"
   PRIMARY KEY(id_mention)
);

-- ROLE
CREATE TABLE ROLE(
   id_role VARCHAR2(20),
   libelle_role VARCHAR2(50),
   description VARCHAR2(255),
   PRIMARY KEY(id_role)
);

-- TYPE_BAC
CREATE TABLE TYPE_BAC(
   id_type VARCHAR2(10),
   libelle_type VARCHAR2(100), -- Exemple: "Général"
   PRIMARY KEY(id_type)
);
```

```sql
-- ENSEIGNANT (Hérite de Utilisateur)
CREATE TABLE ENSEIGNANT(
   id_enseignant NUMBER(10),
   id_role VARCHAR2(20),
   id_utilisateur NUMBER(10) NOT NULL,
   PRIMARY KEY(id_enseignant),
   UNIQUE(id_utilisateur),
   FOREIGN KEY(id_role) REFERENCES ROLE(id_role),
   FOREIGN KEY(id_utilisateur) REFERENCES UTILISATEUR(id_utilisateur)
);

-- GROUPE
CREATE TABLE GROUPE(
   id_groupe VARCHAR2(20), -- Exemple: '3D'
   lettre_groupe CHAR(1), -- Exemple; 'D'
   chiffre_semestre NUMBER(1), -- Exemple: 3
   effectif NUMBER(3),         -- Exemple: 26
   annee_promo NUMBER(4),      -- Exemple: 2025
   id_parcours VARCHAR2(10) NOT NULL,
   PRIMARY KEY(id_groupe),
   FOREIGN KEY(id_parcours) REFERENCES PARCOURS(id_parcours)
);

-- ETUDIANT (Hérite de Utilisateur)
CREATE TABLE ETUDIANT(
   id_etudiant NUMBER(10),
   id_covoiturage NUMBER(10), -- Nullable car pas forcément de groupe covoit
   est_redoublant NUMBER(1) DEFAULT 0 CHECK (est_redoublant IN (0, 1)),
   est_anglophone NUMBER(1) DEFAULT 0 CHECK (est_anglophone IN (0, 1)),
   est_apprenti NUMBER(1) DEFAULT 0 CHECK (est_apprenti IN (0, 1)),
   id_type VARCHAR2(10) NOT NULL,
   id_mention VARCHAR2(10) NOT NULL,
   id_parcours VARCHAR2(10) NOT NULL,
   id_groupe VARCHAR2(20), -- Nullable car affecté plus tard
   id_utilisateur NUMBER(10) NOT NULL,
   PRIMARY KEY(id_etudiant),
   UNIQUE(id_utilisateur),
   FOREIGN KEY(id_type) REFERENCES TYPE_BAC(id_type),
   FOREIGN KEY(id_mention) REFERENCES MENTION_BAC(id_mention),
   FOREIGN KEY(id_parcours) REFERENCES PARCOURS(id_parcours),
   FOREIGN KEY(id_groupe) REFERENCES GROUPE(id_groupe),
   FOREIGN KEY(id_utilisateur) REFERENCES UTILISATEUR(id_utilisateur)
```

```
);

-- NOTE
CREATE TABLE NOTE(
  id_note NUMBER(10),
  valeur_note NUMBER(4,2) NOT NULL, -- Permet 14.50, 20.00...
  commentaire_note VARCHAR2(255),
  id_etudiant NUMBER(10) NOT NULL,
  id_matiere VARCHAR2(20) NOT NULL,
  PRIMARY KEY(id_note),
  FOREIGN KEY(id_etudiant) REFERENCES ETUDIANT(id_etudiant),
  FOREIGN KEY(id_matiere) REFERENCES MATIERE(id_matiere),
  CONSTRAINT chk_note_valide CHECK (valeur_note BETWEEN 0 AND 20)
);

-- ASSOCIATIONS (Tables d'association n,n)
CREATE TABLE ETUDIANT_REPONSE(
  id_etudiant NUMBER(10),
  id_reponse NUMBER(10),
  PRIMARY KEY(id_etudiant, id_reponse),
  FOREIGN KEY(id_etudiant) REFERENCES ETUDIANT(id_etudiant),
  FOREIGN KEY(id_reponse) REFERENCES REPONSE(id_reponse)
);

CREATE TABLE ENSEIGNANT_MATIERE(
  id_enseignant NUMBER(10),
  id_matiere VARCHAR2(20),
  PRIMARY KEY(id_enseignant, id_matiere),
  FOREIGN KEY(id_enseignant) REFERENCES ENSEIGNANT(id_enseignant),
  FOREIGN KEY(id_matiere) REFERENCES MATIERE(id_matiere)
);

CREATE TABLE MATIERE_GROUPE(
  id_groupe VARCHAR2(20),
  id_matiere VARCHAR2(20),
  PRIMARY KEY(id_groupe, id_matiere),
  FOREIGN KEY(id_groupe) REFERENCES GROUPE(id_groupe),
  FOREIGN KEY(id_matiere) REFERENCES MATIERE(id_matiere)
);
```

Partie II : Jeu de test et population de la base de données

Script de population de la base :

C'est un script qui permet d'insérer les données pour pouvoir tester ensuite nos vues, fonctions, requêtes, triggers, procédures. L'ordre qu'on a fait est important par rapport aux clés étrangères.

```
/* ====================
   SCRIPT DE POPULATION
   ==================== */

-- 1. INSERTION DES ROLES
INSERT INTO ROLE (id_role, libelle_role, description) VALUES ('RESP_FORM', 'Responsable Formation', 'Droits totaux sur la plateforme');
INSERT INTO ROLE (id_role, libelle_role, description) VALUES ('RESP_ANNEE', 'Responsable Année/Filière', 'Gestion des groupes et importation des notes');
INSERT INTO ROLE (id_role, libelle_role, description) VALUES ('ENS', 'Enseignant', 'Consultation des groupes et infos pédagogiques');

-- 2. INSERTION DE LA FORMATION
INSERT INTO FORMATION (id_formation, nom_formation) VALUES ('BUT_INFO', 'BUT Informatique');

-- 3. INSERTION DES PARCOURS
INSERT INTO PARCOURS (id_parcours, initiale_parcours, nom_parcours, type_parcours, id_formation) VALUES ('P_A', 'A', 'Développement (Réalisation d''app)', 'Initial', 'BUT_INFO');
INSERT INTO PARCOURS (id_parcours, initiale_parcours, nom_parcours, type_parcours, id_formation) VALUES ('P_B', 'B', 'Cybersécurité (Déploiement)', 'Alternance', 'BUT_INFO');
INSERT INTO PARCOURS (id_parcours, initiale_parcours, nom_parcours, type_parcours, id_formation) VALUES ('P_C', 'C', 'Base de Données (Admin données)', 'Initial', 'BUT_INFO');

-- 4. INSERTION DES TYPES DE BAC ET MENTIONS
INSERT INTO TYPE_BAC (id_type, libelle_type) VALUES ('GEN', 'Bac Général');
INSERT INTO TYPE_BAC (id_type, libelle_type) VALUES ('STI2D', 'Bac Techno STI2D');
INSERT INTO TYPE_BAC (id_type, libelle_type) VALUES ('STMG', 'Bac Techno STMG');
INSERT INTO TYPE_BAC (id_type, libelle_type) VALUES ('PRO', 'Bac Pro SN');

INSERT INTO MENTION_BAC (id_mention, libelle_mention) VALUES ('P', 'Passable');
INSERT INTO MENTION_BAC (id_mention, libelle_mention) VALUES ('AB', 'Assez Bien');
INSERT INTO MENTION_BAC (id_mention, libelle_mention) VALUES ('B', 'Bien');
INSERT INTO MENTION_BAC (id_mention, libelle_mention) VALUES ('TB', 'Très Bien');
```

```
-- 5. INSERTION DES MATIERES
-- Format ID: R + Semestre + . + Numero (exemple : R3.01)
-- lettre_matiere : 'R' (Ressource) ou 'S' (SAÉ/Projet)
INSERT INTO MATIERE (id_matiere, lettre_matiere, numero_matiere, nom_matiere) VALUES
('R3.01', 'R', '301', 'Développement Web');
INSERT INTO MATIERE (id_matiere, lettre_matiere, numero_matiere, nom_matiere) VALUES
('R3.07', 'R', '307', 'SQL et Base de Données');
INSERT INTO MATIERE (id_matiere, lettre_matiere, numero_matiere, nom_matiere) VALUES
('R3.02', 'R', '302', 'Droit des contrats');
INSERT INTO MATIERE (id_matiere, lettre_matiere, numero_matiere, nom_matiere) VALUES
('R3.04', 'R', '304', 'Qualité de développement');
INSERT INTO MATIERE (id_matiere, lettre_matiere, numero_matiere, nom_matiere) VALUES
('R3.14', 'R', '314', 'Anglais Technique'); -- Anciennement ANG
INSERT INTO MATIERE (id_matiere, lettre_matiere, numero_matiere, nom_matiere) VALUES
('R3.15', 'R', '315', 'Mathématiques - Probabilités'); -- Anciennement MATH
INSERT INTO MATIERE (id_matiere, lettre_matiere, numero_matiere, nom_matiere) VALUES
('S3.01', 'S', '301', 'Développement d''application (SAE)'); -- Ajout d'un projet SAE

-- 6. INSERTION DES SONDAGES ET REPONSES
INSERT INTO SONDAGE (id_sondage, nom_sondage, contenu_sondage) VALUES (1,
'Préférences Options S4', 'Classez vos préférences pour les modules complémentaires du
semestre 4.');
INSERT INTO SONDAGE (id_sondage, nom_sondage, contenu_sondage) VALUES (2, 'Besoins
en matériel', 'Avez-vous besoin d''un prêt d''ordinateur portable ?');

INSERT INTO REPONSE (id_reponse, contenu_reponse, id_sondage) VALUES (10, 'IA et Data
Mining', 1);
INSERT INTO REPONSE (id_reponse, contenu_reponse, id_sondage) VALUES (11, 'Réalité
Virtuelle', 1);
INSERT INTO REPONSE (id_reponse, contenu_reponse, id_sondage) VALUES (12,
'Management de projet agile', 1);
INSERT INTO REPONSE (id_reponse, contenu_reponse, id_sondage) VALUES (20, 'Oui', 2);
INSERT INTO REPONSE (id_reponse, contenu_reponse, id_sondage) VALUES (21, 'Non', 2);

-- 7. INSERTION DES UTILISATEURS (ENSEIGNANTS)
INSERT INTO UTILISATEUR VALUES (1, 'Nicolas', 'Ferey', 'nicolas.ferey@univ.fr',
'0102030405', 'Orsay', 'H', 'ENS', TO_DATE('1980-05-12', 'YYYY-MM-DD'), 'nferey', 'hash123');
INSERT INTO UTILISATEUR VALUES (2, 'Jean', 'Dupont', 'jean.dupont@univ.fr', '0102030406',
'Paris', 'H', 'ENS', TO_DATE('1975-08-20', 'YYYY-MM-DD'), 'jdupont', 'hash456');
INSERT INTO UTILISATEUR VALUES (3, 'Marie', 'Curie', 'marie.curie@univ.fr', '0102030407',
'Gif', 'F', 'ENS', TO_DATE('1982-11-03', 'YYYY-MM-DD'), 'mcurie', 'hash789');
```

```
INSERT INTO UTILISATEUR VALUES (4, 'Alan', 'Turing', 'alan.turing@univ.fr', '0102030408',
'Londres', 'H', 'ENS', TO_DATE('1990-01-15', 'YYYY-MM-DD'), 'aturing', 'hash000');
INSERT INTO UTILISATEUR VALUES (5, 'Ada', 'Lovelace', 'ada.lovelace@univ.fr', '0102030409',
'Paris', 'F', 'ENS', TO_DATE('1985-06-30', 'YYYY-MM-DD'), 'alovelace', 'hash111');

-- 8. INSERTION DES ENSEIGNANTS
INSERT INTO ENSEIGNANT (id_enseignant, id_role, id_utilisateur) VALUES (1, 'RESP_FORM',
1);
INSERT INTO ENSEIGNANT (id_enseignant, id_role, id_utilisateur) VALUES (2,
'RESP_ANNEE', 2);
INSERT INTO ENSEIGNANT (id_enseignant, id_role, id_utilisateur) VALUES (3, 'ENS', 3);
INSERT INTO ENSEIGNANT (id_enseignant, id_role, id_utilisateur) VALUES (4, 'ENS', 4);
INSERT INTO ENSEIGNANT (id_enseignant, id_role, id_utilisateur) VALUES (5, 'ENS', 5);

-- 9. INSERTION DES GROUPES
-- Format ID: Chiffre Semestre + Lettre
INSERT INTO GROUPE VALUES ('3A', 'A', 3, 26, 2024, 'P_A');
INSERT INTO GROUPE VALUES ('3B', 'B', 3, 24, 2024, 'P_B');
INSERT INTO GROUPE VALUES ('3C', 'C', 3, 15, 2024, 'P_C');
INSERT INTO GROUPE VALUES ('3D', 'D', 3, 25, 2024, 'P_A');

-- 10. INSERTION DES UTILISATEURS (ETUDIANTS)
INSERT INTO UTILISATEUR VALUES (10, 'Thomas', 'Anderson', 'neo@etu.univ.fr',
'0601010101', 'Zion', 'H', 'ETU', TO_DATE('2004-01-01', 'YYYY-MM-DD'), 'tanderson', 'pass1');
INSERT INTO UTILISATEUR VALUES (11, 'Trinity', 'Moss', 'trinity@etu.univ.fr', '0601010102',
'Zion', 'F', 'ETU', TO_DATE('2004-02-14', 'YYYY-MM-DD'), 'tmoss', 'pass2');
INSERT INTO UTILISATEUR VALUES (12, 'Morpheus', 'Dorne', 'morpheus@etu.univ.fr',
'0601010103', 'Nebuchadnezzar', 'H', 'ETU', TO_DATE('2003-05-20', 'YYYY-MM-DD'), 'mdorne',
'pass3');
INSERT INTO UTILISATEUR VALUES (13, 'Luke', 'Skywalker', 'luke@etu.univ.fr', '0601010104',
'Tatooine', 'H', 'ETU', TO_DATE('2004-09-09', 'YYYY-MM-DD'), 'lsky', 'pass4');
INSERT INTO UTILISATEUR VALUES (14, 'Leia', 'Organa', 'leia@etu.univ.fr', '0601010105',
'Alderaan', 'F', 'ETU', TO_DATE('2004-09-09', 'YYYY-MM-DD'), 'lorgana', 'pass5');
INSERT INTO UTILISATEUR VALUES (15, 'Han', 'Solo', 'han@etu.univ.fr', '0601010106',
'Corellia', 'H', 'ETU', TO_DATE('2003-12-01', 'YYYY-MM-DD'), 'hsolo', 'pass6');
INSERT INTO UTILISATEUR VALUES (16, 'Harry', 'Potter', 'harry@etu.univ.fr', '0601010107',
'Londres', 'H', 'ETU', TO_DATE('2004-07-31', 'YYYY-MM-DD'), 'hpotter', 'pass7');
INSERT INTO UTILISATEUR VALUES (17, 'Hermione', 'Granger', 'hermione@etu.univ.fr',
'0601010108', 'Londres', 'F', 'ETU', TO_DATE('2003-09-19', 'YYYY-MM-DD'), 'hgranger', 'pass8');
INSERT INTO UTILISATEUR VALUES (18, 'Ron', 'Weasley', 'ron@etu.univ.fr', '0601010109',
'Terrier', 'H', 'ETU', TO_DATE('2004-03-01', 'YYYY-MM-DD'), 'rweasley', 'pass9');
```

```
INSERT INTO UTILISATEUR VALUES (19, 'Bilbo', 'Sacquet', 'bilbo@etu.univ.fr', '0601010110',
'Comte', 'H', 'ETU', TO_DATE('2002-09-22', 'YYYY-MM-DD'), 'bsacquet', 'pass10');
INSERT INTO UTILISATEUR VALUES (20, 'Frodo', 'Sacquet', 'frodo@etu.univ.fr', '0601010111',
'Comte', 'H', 'ETU', TO_DATE('2004-09-22', 'YYYY-MM-DD'), 'fsacquet', 'pass11');
INSERT INTO UTILISATEUR VALUES (21, 'Sam', 'Gamgie', 'sam@etu.univ.fr', '0601010112',
'Comte', 'H', 'ETU', TO_DATE('2004-05-12', 'YYYY-MM-DD'), 'sgamgie', 'pass12');
INSERT INTO UTILISATEUR VALUES (22, 'Gandalf', 'Gris', 'gandalf@etu.univ.fr', '0601010113',
'Terre du Milieu', 'H', 'ETU', TO_DATE('2000-01-01', 'YYYY-MM-DD'), 'gandalf', 'pass13');
INSERT INTO UTILISATEUR VALUES (23, 'Lara', 'Croft', 'lara@etu.univ.fr', '0601010114',
'Manoir', 'F', 'ETU', TO_DATE('2003-02-14', 'YYYY-MM-DD'), 'lcroft', 'pass14');
INSERT INTO UTILISATEUR VALUES (24, 'Nathan', 'Drake', 'nathan@etu.univ.fr', '0601010115',
'USA', 'H', 'ETU', TO_DATE('2002-08-10', 'YYYY-MM-DD'), 'ndrake', 'pass15');
INSERT INTO UTILISATEUR VALUES (25, 'Ellie', 'Williams', 'ellie@etu.univ.fr', '0601010116',
'Boston', 'F', 'ETU', TO_DATE('2005-01-01', 'YYYY-MM-DD'), 'ewilliams', 'pass16');

-- 11. INSERTION DES ETUDIANTS
INSERT INTO ETUDIANT VALUES (10, 100, 0, 1, 0, 'GEN', 'TB', 'P_A', '3A', 10);
INSERT INTO ETUDIANT VALUES (11, 100, 0, 1, 0, 'GEN', 'B', 'P_A', '3A', 11);
INSERT INTO ETUDIANT VALUES (12, 100, 1, 0, 0, 'STI2D', 'AB', 'P_A', '3A', 12);

INSERT INTO ETUDIANT VALUES (13, NULL, 0, 0, 1, 'GEN', 'TB', 'P_B', '3B', 13);
INSERT INTO ETUDIANT VALUES (14, NULL, 0, 1, 1, 'GEN', 'B', 'P_B', '3B', 14);

INSERT INTO ETUDIANT VALUES (15, NULL, 0, 0, 0, 'STI2D', 'P', 'P_C', '3C', 15);
INSERT INTO ETUDIANT VALUES (16, 101, 0, 1, 0, 'GEN', 'B', 'P_C', '3C', 16);
INSERT INTO ETUDIANT VALUES (17, NULL, 0, 1, 0, 'GEN', 'TB', 'P_C', '3C', 17);
INSERT INTO ETUDIANT VALUES (18, 101, 1, 0, 0, 'PRO', 'AB', 'P_C', '3C', 18);

INSERT INTO ETUDIANT VALUES (19, NULL, 0, 0, 0, 'GEN', 'P', 'P_A', '3D', 19);
INSERT INTO ETUDIANT VALUES (20, NULL, 0, 0, 0, 'GEN', 'AB', 'P_A', '3D', 20);
INSERT INTO ETUDIANT VALUES (21, NULL, 0, 0, 0, 'STMG', 'AB', 'P_A', '3D', 21);
INSERT INTO ETUDIANT VALUES (22, NULL, 0, 1, 0, 'GEN', 'TB', 'P_B', '3B', 22);
INSERT INTO ETUDIANT VALUES (23, NULL, 0, 0, 0, 'GEN', 'B', 'P_B', '3B', 23);
INSERT INTO ETUDIANT VALUES (24, NULL, 0, 0, 0, 'STI2D', 'P', 'P_C', '3C', 24);
INSERT INTO ETUDIANT VALUES (25, NULL, 0, 1, 0, 'GEN', 'B', 'P_A', '3A', 25);

-- 12. INSERTION DES NOTES
-- R3.01 (Web), R3.07 (SQL), R3.14 (Anglais), R3.15 (Maths)
INSERT INTO NOTE VALUES (1, 15.5, 'Bon travail', 10, 'R3.07');
INSERT INTO NOTE VALUES (2, 18.0, 'Excellent', 10, 'R3.01');
INSERT INTO NOTE VALUES (3, 14.0, '', 10, 'R3.14');
INSERT INTO NOTE VALUES (4, 08.5, 'Attention', 12, 'R3.07');
```

```
INSERT INTO NOTE VALUES (5, 10.0, 'Juste', 12, 'R3.01');
INSERT INTO NOTE VALUES (6, 19.5, 'Parfait', 17, 'R3.07');
INSERT INTO NOTE VALUES (7, 16.0, '', 17, 'R3.15');
INSERT INTO NOTE VALUES (8, 12.0, '', 11, 'R3.07');
INSERT INTO NOTE VALUES (9, 11.5, '', 13, 'R3.01');
INSERT INTO NOTE VALUES (10, 13.0, '', 14, 'R3.02');
INSERT INTO NOTE VALUES (11, 09.0, 'A revoir', 15, 'R3.15');
INSERT INTO NOTE VALUES (12, 17.0, 'Very good', 16, 'R3.14');
INSERT INTO NOTE VALUES (13, 05.0, 'Absent', 19, 'R3.07');
INSERT INTO NOTE VALUES (14, 14.5, '', 20, 'R3.04');
-- Note sur le projet SAE
INSERT INTO NOTE VALUES (15, 16.0, 'Bonne soutenance', 10, 'S3.01');

-- 13. INSERTION DANS LES TABLES D'ASSOCIATION
-- Table ETUDIANT_REPONSE
INSERT INTO ETUDIANT_REPONSE VALUES (10, 10);
INSERT INTO ETUDIANT_REPONSE VALUES (11, 11);
INSERT INTO ETUDIANT_REPONSE VALUES (12, 20);

-- Table ENSEIGNANT_MATIERE
INSERT INTO ENSEIGNANT_MATIERE VALUES (1, 'R3.07');
INSERT INTO ENSEIGNANT_MATIERE VALUES (1, 'R3.01');
INSERT INTO ENSEIGNANT_MATIERE VALUES (3, 'R3.15');

-- Table MATIERE_GROUPE
INSERT INTO MATIERE_GROUPE VALUES ('3A', 'R3.07');
INSERT INTO MATIERE_GROUPE VALUES ('3A', 'R3.01');
INSERT INTO MATIERE_GROUPE VALUES ('3B', 'R3.04');
INSERT INTO MATIERE_GROUPE VALUES ('3C', 'R3.07');

COMMIT;
```

Réalisation des requêtes et des vues associées

Les vues ont été créées pour simplifier les requêtes répétitives dans l'application (Java/PHP).

- V_INFO_COMPLETE_ETUDIANT : Cette vue centralise l'ensemble des informations (administratives et scolaires) d'un étudiant en réalisant une jointure entre 5 tables.

Cela évite de réécrire des jointures complexes à chaque fois que l'application doit afficher une liste d'étudiants.

- V_DASHBOARD_GROUPES : Vue analytique qui agrège les statistiques par groupe. Elle calcule dynamiquement le nombre d'inscrits, le nombre de femmes, de redoublants et d'apprentis. Cela permet d'alimenter le tableau de bord de constitution des groupes (en temps réel), permettant au responsable de vérifier instantanément le respect des contraintes (Mixité, Effectifs, Hétérogénéité).
- V_MOYENNE_ETUDIANT : Cette vue qui calcule la moyenne générale actuelle de chaque étudiant en se basant sur toutes ses notes enregistrées Elle permet de classer les étudiants pour équilibrer les groupes.
- V_RESULTATS_SONDAGE : Cette vue fait le lien entre les étudiants et leurs réponses aux sondages, ce qui facilite la création de groupes par affinité.

Les requêtes correspondent aux fonctionnalités clés identifiées dans les maquettes et les cas d'utilisation. Les requêtes permettent de tester notre base de données et d'analyser de potentielles erreurs.

- Identification des étudiants sans affectation : cette requête sélectionne les étudiants depuis la vue V_INFO_COMPLETE_ETUDIANT dont le champ id_groupe est NULL.
- Contrôle de l'équilibre académique : cette requête calcule la moyenne des moyennes de chaque groupe V_DASHBOARD_GROUPES pour trouver les groupes où le nombre de femmes est égal à 0 ou égal au nombre total d'inscrits (donc 0 homme). L'objectif est de lever une alerte si la contrainte de diversité des genres n'est pas respectée.
- Filtrage par critère de sondage : cette requête récupère les identités des étudiants ayant donné une réponse spécifique (exemple: 'IA et Data Mining'). Cela permet au responsable de sélectionner un ensemble d'étudiants partageant un même intérêt.

```
/* =========
   LES VUES
   ========= */

/*
   Utilisée pour "Mes informations" et "Liste des étudiants"
   Scénarios : 1 (Étudiant) et 10 (Enseignant)
   Objectif : Avoir toutes les infos (Nom, Prénom, Bac, Groupe, Parcours) sans faire 5 jointures à chaque fois
*/
CREATE OR REPLACE VIEW V_INFO_COMPLETE_ETUDIANT AS
SELECT
    e.id_etudiant,
    u.nom_utilisateur,
    u.prenom_utilisateur,
    u.mail_utilisateur,
    u.genre_utilisateur,
    u.statut_utilisateur,
```

```
    tb.libelle_type AS type_bac,
    mb.libelle_mention AS mention_bac,
    p.nom_parcours,
    e.est_redoublant,
    e.est_apprenti,
    g.id_groupe,
    g.lettre_groupe
FROM ETUDIANT e
JOIN UTILISATEUR u ON e.id_utilisateur = u.id_utilisateur
JOIN TYPE_BAC tb ON e.id_type = tb.id_type
LEFT JOIN MENTION_BAC mb ON e.id_mention = mb.id_mention
JOIN PARCOURS p ON e.id_parcours = p.id_parcours
LEFT JOIN GROUPE g ON e.id_groupe = g.id_groupe;


/*
   Utilisée pour "Constitution des groupes"
   Scénarios : 7 (Répartition automatique)
   Objectif : Visualiser en temps réel si les contraintes (Mixité, Effectif, Redoublant....) sont
respectées
*/
CREATE OR REPLACE VIEW V_DASHBOARD_GROUPES AS
SELECT
    g.id_groupe,
    g.lettre_groupe,
    g.effectif AS capacite_max,
    COUNT(e.id_etudiant) AS nb_inscrits,
    -- Calcul du nombre de filles pour la contrainte de mixité
    SUM(CASE WHEN u.genre_utilisateur = 'F' THEN 1 ELSE 0 END) AS nb_femmes,
    -- Calcul du nombre de redoublants pour l'homogénéité
    SUM(CASE WHEN e.est_redoublant = 1 THEN 1 ELSE 0 END) AS nb_redoublants,
    -- Calcul du nombre d'apprentis
    SUM(CASE WHEN e.est_apprenti = 1 THEN 1 ELSE 0 END) AS nb_apprentis
FROM GROUPE g
LEFT JOIN ETUDIANT e ON g.id_groupe = e.id_groupe
LEFT JOIN UTILISATEUR u ON e.id_utilisateur = u.id_utilisateur
GROUP BY g.id_groupe, g.lettre_groupe, g.effectif;


/*
   Algorithme de répartition.
   Scénarios : 7 (Règle "Équilibrer les niveaux académiques" )
```

```
   Objectif : Calculer la moyenne générale de chaque étudiant pour pouvoir les trier par niveau
*/
CREATE OR REPLACE VIEW V_MOYENNE_ETUDIANT AS
SELECT
    e.id_etudiant,
    u.nom_utilisateur,
    u.prenom_utilisateur,
    AVG(n.valeur_note) AS moyenne_generale
FROM ETUDIANT e
JOIN UTILISATEUR u ON e.id_utilisateur = u.id_utilisateur
JOIN NOTE n ON e.id_etudiant = n.id_etudiant
GROUP BY e.id_etudiant, u.nom_utilisateur, u.prenom_utilisateur;



/*
   Aide à la décision manuelle.
   Scénarios : 3 (Réponse sondage) et 7 (Création groupe)
   Objectif : Voir qui a répondu quoi pour regrouper par choixx (exemple : "Options" ou
"Covoiturage")
*/
CREATE OR REPLACE VIEW V_RESULTATS_SONDAGE AS
SELECT
    s.nom_sondage,
    r.contenu_reponse,
    u.nom_utilisateur,
    u.prenom_utilisateur,
    e.id_etudiant
FROM SONDAGE s
JOIN REPONSE r ON s.id_sondage = r.id_sondage
JOIN ETUDIANT_REPONSE er ON r.id_reponse = er.id_reponse
JOIN ETUDIANT e ON er.id_etudiant = e.id_etudiant
JOIN UTILISATEUR u ON e.id_utilisateur = u.id_utilisateur;



/* =========
   LES REQUETES
   ========= */
/*
   Liste des étudiants sans groupe
   Fonctionnalité : Répartition manuelle ou Initialisation de l'algo
   Maquette : "Groupes à constituer"
*/
```

```sql
SELECT id_etudiant, nom_utilisateur, prenom_utilisateur, nom_parcours
FROM V_INFO_COMPLETE_ETUDIANT
WHERE id_groupe IS NULL
ORDER BY nom_utilisateur;


/* Vérification de l'équilibre des niveaux académiques par groupe
   Fonctionnalité : Vérification après répartition automatique
   Objectif : Comparer la moyenne de chaque groupe pour voir s'ils sont équilibrés
*/
SELECT
    g.lettre_groupe,
    ROUND(AVG(vme.moyenne_generale), 2) AS moyenne_du_groupe
FROM GROUPE g
JOIN ETUDIANT e ON g.id_groupe = e.id_groupe
JOIN V_MOYENNE_ETUDIANT vme ON e.id_etudiant = vme.id_etudiant
GROUP BY g.lettre_groupe
ORDER BY moyenne_du_groupe DESC;


/* Détection des groupes non mixtes
   Fonctionnalité : Contrainte de mixité
   Objectif : Identifier les groupes qui n'ont aucune femme ou aucun homme.
*/
SELECT id_groupe, lettre_groupe, nb_femmes, (nb_inscrits - nb_femmes) as nb_hommes
FROM V_DASHBOARD_GROUPES
WHERE nb_femmes = 0
   OR (nb_inscrits - nb_femmes) = 0;


/* Récupérer les étudiants ayant choisi une option spécifique pour les mettre ensemble
   Fonctionnalité : Regroupement par critère
   Objectif : Trouver tous les étudiants ayant répondu "IA et Data Mining" au sondage
*/
SELECT e.id_etudiant, u.nom_utilisateur, u.prenom_utilisateur
FROM ETUDIANT e
JOIN UTILISATEUR u ON e.id_utilisateur = u.id_utilisateur
JOIN ETUDIANT_REPONSE er ON e.id_etudiant = er.id_etudiant
JOIN REPONSE r ON er.id_reponse = r.id_reponse
WHERE r.contenu_reponse = 'IA et Data Mining';

COMMIT;
```

Partie III : Réalisation/implémentation de la base de données, aspect dynamique

Implémenter les triggers et leurs fonctions PL/SQL associées

Les fonctions implémentent des calculs souvent utilisés par l'application.

- F_CALCUL_MOYENNE : Calcule la moyenne générale d'un étudiant donné sur l'ensemble de ses notes. Elle gère spécifiquement le cas des étudiants sans note en retournant -1.
- F_EST_GROUPE_COMPLET : Vérifie si le nombre d'étudiants inscrits dans un groupe a atteint ou dépassé son effectif maximum défini. Retourne un booléen (1 ou 0).
- F_TAUX_FEMMES_GROUPE : Calcule le pourcentage de femmes dans un groupe spécifique via une jointure entre ETUDIANT et UTILISATEUR (pour accéder au genre), permettant de visualiser le respect de la contrainte de mixité.

Les procédures encapsulent des traitements impliquant souvent plusieurs étapes ou des modifications de données (INSERT/UPDATE).

- P_AFFECTER_ETUDIANT : Gère l'inscription d'un étudiant dans un groupe, en appelant la fonction F_EST_GROUPE_COMPLET pour vérifier la capacité, si le groupe est plein, elle lève une exception, sinon, elle procède à la mise à jour.
- P_RESET_GROUPES_PARCOURS : Réinitialise l'affectation de tous les étudiants d'un parcours donné. (SET id_groupe = NULL)
- P_AJOUTER_NOTE : Facilite l'insertion de nouvelles notes en gérant automatiquement la clé primaire via une séquence. Elle inclut une gestion d'erreur pour éviter les doublons.

Les triggers assurent l'automatisation et la sécurité des données de manière transparente pour l'application.

- TRG_FORMAT_UTILISATEUR : Formate automatiquement les données d'identité avec le nom en majuscules, le prénom avec la première lettre en majuscule et l’email en minuscules.
- TRG_CHECK_COHERENCE_PARCOURS : Intercepte chaque affectation de groupe pour vérifier que le parcours du groupe correspond bien au parcours de l'étudiant. Si incompatibilité, l'opération est bloquée.
- TRG_AUDIT_NOTE : À chaque modification d'une note, ce trigger insère une ligne dans la table d'historique LOG_MODIF_NOTE contenant l'ancienne valeur, la nouvelle valeur, la date et l'utilisateur responsable.

```sql
SET SERVEROUTPUT ON;

/* ==============
   LES FONCTIONS
   ============== */

/*
   Calculer la moyenne générale d'un étudiant
   Objectif : Servira pour l'algorithme de répartition par niveau académique
   Retourne : La moyenne sur 20 ou -1 si aucune note
*/
CREATE OR REPLACE FUNCTION F_CALCUL_MOYENNE(p_id_etudiant IN NUMBER)
RETURN NUMBER
IS
    v_moyenne NUMBER(4,2);
    v_count NUMBER;
BEGIN
    -- Vérifier si l'étudiant a des notes
    SELECT COUNT(*) INTO v_count FROM NOTE WHERE id_etudiant = p_id_etudiant;

    IF v_count = 0 THEN
        RETURN -1; -- "Pas de note"
    ELSE
        SELECT AVG(valeur_note) INTO v_moyenne
        FROM NOTE
        WHERE id_etudiant = p_id_etudiant;
        RETURN ROUND(v_moyenne, 2);
    END IF;
END F_CALCUL_MOYENNE;
/

/*
   Vérifier si un groupe a atteint sa capacité maximale
   Objectif : Empêcher la surcharge des groupes lors de la répartition
   Retourne : 1 vrai donc plein ou 0 faux donc place dispo
*/
CREATE OR REPLACE FUNCTION F_EST_GROUPE_COMPLET(p_id_groupe IN VARCHAR2)
RETURN NUMBER
IS
    v_nb_inscrits NUMBER;
    v_capacite NUMBER;
BEGIN
```

```
    SELECT effectif INTO v_capacite FROM GROUPE WHERE id_groupe = p_id_groupe;

    SELECT COUNT(*) INTO v_nb_inscrits FROM ETUDIANT WHERE id_groupe =
p_id_groupe;

    IF v_nb_inscrits >= v_capacite THEN
        RETURN 1; -- groupe plein
    ELSE
        RETURN 0; -- il reste de la place
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RETURN 1; -- bloque par securite si le groupe n'existe pas
END F_EST_GROUPE_COMPLET;
/

/*
   Calculer le taux de féminisation d'un groupe
   Objectif : Aider à respecter la contrainte de mixité
   Retourne : Le pourcentage de femmes dans le groupe
*/
CREATE OR REPLACE FUNCTION F_TAUX_FEMMES_GROUPE(p_id_groupe IN
VARCHAR2)
RETURN NUMBER
IS
    v_total NUMBER;
    v_nb_femmes NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_total FROM ETUDIANT WHERE id_groupe = p_id_groupe;

    IF v_total = 0 THEN
        RETURN 0;
    END IF;

    SELECT COUNT(*) INTO v_nb_femmes
    FROM ETUDIANT e
    JOIN UTILISATEUR u ON e.id_utilisateur = u.id_utilisateur
    WHERE e.id_groupe = p_id_groupe AND u.genre_utilisateur = 'F';

    RETURN ROUND((v_nb_femmes / v_total) * 100, 2);
END F_TAUX_FEMMES_GROUPE;
/
```

```
/* ================
   LES PROCEDURES
   ============== */

/*
   Affectation manuelle sécurisée
   Objectif : Ajouter un étudiant à un groupe en vérifiant les contraintes
   Lève une erreur si l'action est impossible
*/
CREATE OR REPLACE PROCEDURE P_AFFECTER_ETUDIANT(
    p_id_etudiant IN NUMBER,
    p_id_groupe IN VARCHAR2
)
IS
    v_est_complet NUMBER;
BEGIN
    v_est_complet := F_EST_GROUPE_COMPLET(p_id_groupe);

    IF v_est_complet = 1 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Erreur : Le groupe ' || p_id_groupe || ' est
complet.');
    END IF;

    UPDATE ETUDIANT
    SET id_groupe = p_id_groupe
    WHERE id_etudiant = p_id_etudiant;

    IF SQL%ROWCOUNT = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Erreur : Étudiant introuvable.');
    END IF;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Succès : Étudiant ' || p_id_etudiant || ' affecté au groupe ' ||
p_id_groupe);
END P_AFFECTER_ETUDIANT;
/

/*
   Réinitialisation des groupes d'un parcours
   Objectif : Permet de remettre à zéro la répartition avant de relancer un algo automatique
   Correspond au besoin d'ajustements fréquents
```

```
*/
CREATE OR REPLACE PROCEDURE P_RESET_GROUPES_PARCOURS(p_id_parcours IN
VARCHAR2)
IS
BEGIN
    UPDATE ETUDIANT
    SET id_groupe = NULL
    WHERE id_parcours = p_id_parcours;

    COMMIT;
    DBMS_OUTPUT.PUT_LINE('Les groupes du parcours ' || p_id_parcours || ' ont été
réinitialisés.');
END P_RESET_GROUPES_PARCOURS;
/

/*
   Importation unitaire de note (Simulation d'un imporrt CSV)
   Objectif : Facilite l'insertion des notes importées depuis un fichier
   Gère la création automatique si la note n'existe pas
*/

-- Pré-requis :
CREATE SEQUENCE SEQ_ID_NOTE
START WITH 1
INCREMENT BY 1;

CREATE OR REPLACE PROCEDURE P_AJOUTER_NOTE(
    p_id_etudiant IN NUMBER,
    p_id_matiere IN VARCHAR2,
    p_valeur IN NUMBER,
    p_commentaire IN VARCHAR2
)
IS
BEGIN
    INSERT INTO NOTE (id_note, valeur_note, commentaire_note, id_etudiant, id_matiere)
    VALUES (SEQ_ID_NOTE.NEXTVAL, p_valeur, p_commentaire, p_id_etudiant, p_id_matiere);
    -- Note: Suppose l'existence d'une séquence SEQ_ID_NOTE

    COMMIT;
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Erreur : Une note existe déjà pour cette matière/étudiant.');
```

```
END P_AJOUTER_NOTE;
/


/* ==============
   LES TRIGGERS
   ============== */
/*
   Formatage automatique des données utilisateur
   Objectif : Garantir la propreté des données (Nom en maj, email en minuscule) à l'inscription
   Type : BEFORE INSERT OR UPDATE
*/
CREATE OR REPLACE TRIGGER TRG_FORMAT_UTILISATEUR
BEFORE INSERT OR UPDATE ON UTILISATEUR
FOR EACH ROW
BEGIN
    :NEW.nom_utilisateur := UPPER(:NEW.nom_utilisateur);
    :NEW.prenom_utilisateur := INITCAP(:NEW.prenom_utilisateur);
    :NEW.mail_utilisateur := LOWER(:NEW.mail_utilisateur);
END;
/

/*
   Vérification de cohérence Parcours / Groupe
   Objectif : Un étudiant inscrit en parcours A ne doit pas être mis dans un groupe du parcours B
   Type : BEFORE UPDATE
*/
CREATE OR REPLACE TRIGGER TRG_CHECK_COHERENCE_PARCOURS
BEFORE UPDATE OF id_groupe ON ETUDIANT
FOR EACH ROW
WHEN (NEW.id_groupe IS NOT NULL) -- que si on affecte un groupe
DECLARE
    v_parcours_groupe VARCHAR2(10);
BEGIN
    SELECT id_parcours INTO v_parcours_groupe
    FROM GROUPE
    WHERE id_groupe = :NEW.id_groupe;

    IF v_parcours_groupe != :NEW.id_parcours THEN
        RAISE_APPLICATION_ERROR(-20003,
            'Incohérence : Ce groupe appartient au parcours ' || v_parcours_groupe ||
            ' alors que l''étudiant est inscrit en ' || :NEW.id_parcours);
```

```
    END IF;
END;
/
/*
   Audit des modifications de notes
   Objectif : Garder une trace si un enseignant modifie une note
   Nécessite une table TABLE_LOG_NOTE
*/
-- Pré-requis :
-- Création de la table LOG
CREATE TABLE LOG_MODIF_NOTE (
   id_log NUMBER(10),
   ancien_valeur NUMBER(4,2),
   nouvelle_valeur NUMBER(4,2),
   date_modif DATE DEFAULT SYSDATE,
   user_modif VARCHAR2(50),
   id_note_concernee NUMBER(10),
   PRIMARY KEY(id_log)
);

-- Création de la séquence pour générer les ID automatiquemnt
CREATE SEQUENCE SEQ_LOG_NOTE
START WITH 1
INCREMENT BY 1;

CREATE OR REPLACE TRIGGER TRG_AUDIT_NOTE
AFTER UPDATE OF valeur_note ON NOTE
FOR EACH ROW
BEGIN
    INSERT INTO LOG_MODIF_NOTE (id_log, ancien_valeur, nouvelle_valeur, date_modif,
user_modif)
    VALUES (
        SEQ_LOG_NOTE.NEXTVAL,
        :OLD.valeur_note,
        :NEW.valeur_note,
        SYSDATE,
        rdahma1 -- L'utilisateur Oracle qui fait la modif
    );
END;
/
```