UNIVERSITE ABOU BEKR BELKAID – TLEMCEN
FACULTE DES SCIENCES - DEPARTEMENT D'INFORMATIQUE

L3 **Administration des BD** Durée : **2H** 09/07/2019

Questions (4 Pts)

1. Quelle est la différence entre jointure interne et jointure externe. **(0.5 Pt)**
   - **Jointure Interne (INNER JOIN)** : Jointure interne ou fermée, les données doivent être à la fois dans les 2 tables (Choix par défaut)
   - **Jointure Externe (OUTER JOIN)** : Jointure externe ou ouverte, on lit les données d'une table en y associant éventuellement celle de l'autre table (Remplir avec des valeurs NULL si la condition n'est pas respectée)
2. Citer les manières et les niveaux de définition de contraintes. (**1Pt)**
   - **3 manières** : Contrainte de colonne, Contrainte de table, Ajout de la contrainte ultérieurement
   - **3 niveaux** : Intra-Table Verticale, Intra-Table Horizontale, Inter-Tables
3. Quels sont les contextes d'utilisations de vue, vue matérialisée et index. (**0,75 Pt)**
   - **Vue** : Confidentialité, Cacher la complexité des données aux utilisateurs, Requêtes complexes, Présenter différentes perspectives sur les données aux utilisateurs, Mise à jour des tables transparente aux utilisateurs
   - **Vue matérialisée** : BDs Distribuées, Entrepôts de données (DataWarehouse)
   - **Index** : Accès rapide et direct aux lignes des tables
4. Lister les étapes de traitement d'une requête ? **(0.75 Pt)**
   - Analyse
   - Compilation
   - Optimisation
   - Exécution
5. Quels sont les principaux algorithmes d'implémentation de jointures ? **(1 Pt)**
   - Jointures par boucles imbriquées (Nested Loop)
   - Jointure avec boucle indexée
   - Jointure par tri fusion (Sort-Join)
   - Jointure par hachage (Hash-Join)

- Soient les deux tables **ETUDIANT**et **PROMOTION** définies comme suit :

  ETUDIANT (CodeEt, Nom, DateNaiss, Commune, Promotion#)
  PROMOTION (CodeProm,Désignation, Département, Année, Délégué#)

Partie 1 - Donner les requêtes SQL suivantes (9 Pts) :

1. Créer les deux relations de la base de données. **(1 Pt)**

```
CREATE TABLE ETUDIANT (
CodeEt NUMBER(4) PRIMARY KEY,
Nom VARCHAR(20),
DateNaissDATE,
Commune VARCHAR(15),
Promotion VARCHAR(4) ) ; 0,25

CREATE TABLE PROMOTION (
CodeProm VARCHAR(4) PRIMARY KEY,
Designation VARCHAR(20),
Departement VARCHAR(15),
Annee NUMBER(4),
Delegue NUMBER(4), 0,25
CONSTRAINT fk_deleg FOREIGN KEY Delegue References ETUDIANT (CodeEt)) ; 0,25

ALTER TABLE ETUDIANT ADD CONSTRAINT fk_prom FOREIGN KEY Promotion REFERENCES
PROMOTION (CodeProm)) ; 0,25
```

2. Décider, justifier et définir la stratégie adoptée en cas d'abandon(suppression) d'un délégué. **(0.75 Pt)**

Abandon d'un délégué (Suppression d'un étudiant délégué) :

ALTER TABLE PROMOTION ADD CONSTRAINT FK_Prom FOREIGN KEY Delegue REFERENCES ETUDAINT (CodeET) ON DELETE SET NULL [ou SET DEFAULT]; **0,25**

Justification : Si un étudiant délégué abandonne, le SGBD va autoriser la suppression de la table ETUDIANT, avec remplacement de l'attribut concerné dans la table PROMOTION par « NULL » ou « DEFAULT », en conservant très naturellement la promotion avec toutes ses informations en attendant l'élection d'un nouveau délégué. **0,5**

3. Quelle est la solution à choisir en cas ou de réélection d'un nouveau délégué **? (0.75 Pt)**
   Réélection d'un nouveau délégué : Aucune stratégie à adopter, une seule modification dans la table PROMOTION suffit : **0,25**
   UPDATE PROMOTION SET Delegue = NouvDeleg WHERE Delegue = AncDeleg ; **0,5**

4. Définir les contraintes qui imposent que l'attribut Délégué dans la table PROMOTION doit exister toujours et que le couple (Désignation, Département) ne doit pas se répéter. **(0.5Pt)**
   Delegue dans la table PROMOTION est une clé étrangère qui référence l'attribut CodeEt qui est la clé primaire de la table ETUDIANT, NOT NULL est assuré par le Foreign key: inutile de définir cette contrainte.
   ALTER TABLE PROMOTIONADD CONSTRAINT Un_DesDepUNIQUE (Designation, Departement) ;

5. Ajouter la contrainte suivante : l'étudiant délégué d'une promotion doit être inscrit dans cette promotion. **(0.5 Pt)**
   ALTER TABLE PROMOTION ADD CONSTRAINT Ci_Deleg CHECK (Delegue=[ou In] (SELECT CodeEt FROM ETUDIANT WHERE Etudiant.PROMOTION = PROMOTION.CodeProm)) ;

6. Comment limiter l'inscription d'un étudiant à une seule promotion **? (0.5 Pts)**
   Cette règle est déjà prise en charge par la première forme normale (1FN) du modèle relationnel.
   Pour un étudiant : CodeEt→Promotion (1 seule promotion)

7. Est-ce qu'on peut supprimer la promotion « M1_Chimie » composée de 22 étudiants **?** Sinon, quelles sont les différentes solutions proposées pour pouvoir la supprimer **?** (Sans requêtes) (**1 Pt)**
   La réponse : Non, Impossible de supprimer une ligne M1_Chimie de la table PROMOTION qui est référencée par la table ETUDIANT.

   Les solutions possibles à ce problème :
   - Spécifier de stratégie ; **0,25**
     Ou
   - Supprimer d'abord les lignes de la table Etudiant qui font référence à la promotion fournisseur «M1_Chimie» ; **0,25**
     Ou
   - Désactiver la contrainte de clé étrangère ; **0,25**
     Ou
   - Supprimer la contrainte (Mauvaise solution : erreur de conception) ; **0,25**

8. Supprimer les étudiants du département de Maths qui ne sont pas des délégués. **(0.5 Pt)**
   DELETE FROM ETUDIANT WHERE CodeEt NOT IN (SELECT Delegue FROM PROMOTION WHERE Departement='Maths');

9. Afficher tous les noms des étudiants inscrits et non encore inscrits dans une promotion avec les désignations de promotions s'il y en a, en reportant à la fin ceux qui ne sont pas inscrits. **(1 Pt)**
   SELECT Nom, Designation FROM ETUDIANT LEFT JOIN PROMOTION ON. **(0.5 Pt)**
   ETUDIANT.Promotion=PROMOTION.CodeProm ORDER BY Designation NULLS LAST ; **(0.5 Pt)**

10. Créer une vue PH18-19, qui va se limiter à la désignation de la promotion et son code de délégué, des promotionsdu département physiques des années 2018 et 2019, qui interdit l'insertion de nouvelles promotions à travers cette vue. **(1 Pt)**
    CREATE VIEW PH18-19 **(0.25 Pt)**
    AS SELECT Designation, Delegue FROM PROMOTION
    WHERE (Departement ='Physiques') AND (Annee='2018' OR Annee='2019') **(0.25 Pt)**
    WITH READ ONLY **(0.5 Pt)**

11. Créer une réplique DELEGUE, contenant les noms des délégués avec les désignations de promotions correspondantes, qui se renseigne au prochain rafraîchissement, et qui est mise à jour mensuellement. **(1.5 Pt)**
    CREATE MATERIALIZED VIEW DELEGUE **(0.25 Pt)**
    BUILD Deferred **(0.5 Pt)**
    REFRESH START Sysdate NEXT Sysdate+30 **(0.5 Pt)**
    AS SELECT Designation, Nom FROM PROMOTION, ETUDIANT
    WHERE PROMOTION.Delegue = ETUDIANT.CodeEt ; **(0.25 Pt)**

Partie 2- (7 Pts)

1. Donner la liste des noms des étudiants du département d'Informatique, qui sont inscrits dans la même promotion que l'étudiant « C105 ».

   1.1 Donnez la requête SQL. **(1 Pt)**

   SELECT Nom FROM ETUDIANT WHERE Promotion=(SELECT Promotion FROM ETUDIANT WHERE CodeEt='C105') AND Promotion=(SELECT CodeProm FROM PROMOTION WHERE Departement='Informatique';

   Ou

   SELECT Nom FROM ETUDIANT WHERE Promotion=(SELECT Promotion FROM ETUDIANT WHERE CodeEt='C105')

   INTERSECT

   SELECT Nom FROM ETUDIANT WHERE Promotion=(SELECT CodeProm FROM PROMOTION WHERE Departement='Informatique';

   Ou

   SELECT Nom FROM ETUDIANT, PROMOTION WHERE Departement='Informatique' AND CodeProm =(SELECT Promotion FROM ETUDIANT WHERE CodeEt='C105')

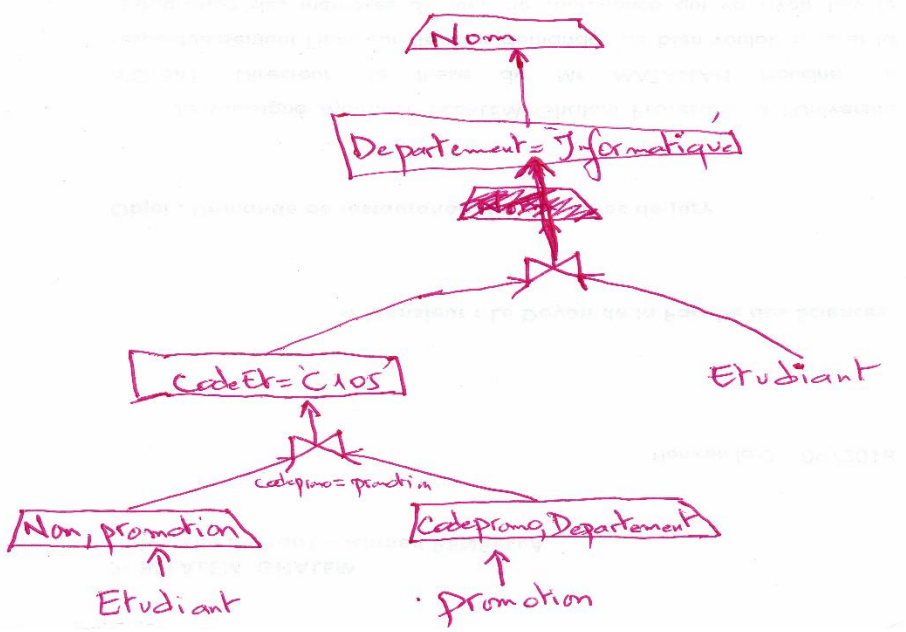
   1.2 Optimiser votre requête SQL. **(1 Pt)**

   SELECT Nom FROM (SELECT Nom, Promotion FROM ETUDIANT) WHERE Promotion=(SELECT Promotion FROM EUDIANT WHERE CodeEt='C105') AND Promotion=(SELECT CodeProm FROM PROMOTION WHERE Departement='Informatique';

   1.3 Traduire la requête SQL optimisée en algèbre relationnelle. **(0,5 Pt)**

   $\pi_{Nom}(\sigma_{Departement='Informatique'}(ETUDIANT \bowtie (\sigma_{CodeEt='C105'}(\pi_{[Nom, Promotion]} ETUDIANT \bowtie \pi_{[CodeProm, Departement]} PROMOTION))))$

   1.4 Schématiser l'arbre algébrique correspondant. **(0,5 Pt)**



   1.5 Proposer le plan d'exécution optimal. **(0,5 Pt)**

   R1← Projeter Etudiant sur Nom et Promotion
   R2← Projeter Promotion sur CodeProm et Departement
   R3← Joindre R1avec R2 par CodeProm=Promotion
   R4← Sélectionner de la table R3 les lignes dont le CodeEt='C105'
   R5← Projeter Etudiant sur Nom
   R6← Joindre R4 avec R5 par CodeProm=Promotion
   R7← Sélectionner de la table R6 les lignes dont le Departement='Informatique'
   R8← Projeter R7 sur Nom

2. Lister les couples de noms d'étudiants de la même promotion résidant la même commune.

2.1 Donnez la requête SQL. **(1 Pt)**

SELECT E1.Nom , E2.NomFROM ETUDIANT AS E1 , ETUDIANT AS E2 WHERE E1.Promotion=E2.Promotion AND E1.Commune=E2.Commune AND E1.CodeEt > E2.CodeEt;

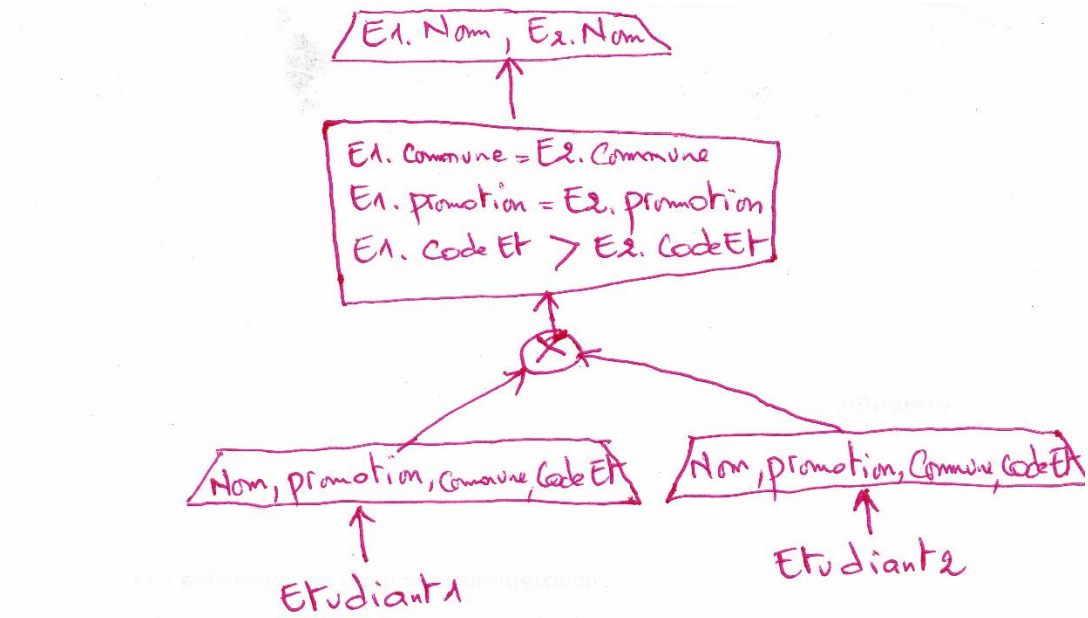
2.2 Traduire la requête SQL en algèbre relationnelle. **(0,5 Pt)**

$\pi_{[E1.Nom,\ E2.Nom]}(\sigma_{E1.Promotion=E2.Promotion\ \wedge\ E1.Commune=E2.Commune\ \wedge\ E1.CodeEt\ >\ E2.CodeEt}(ETUDIANT\ E1 \bowtie ETUDIANT\ E2))$

2.3 Optimiser votre requête en algèbre relationnelle. **(1 Pt)**

$\pi_{[E1.Nom, E2.Nom]}(\sigma_{E1.Promotion=E2.Promotion \wedge E1.Commune=E2.Commune \wedge E1.CodeEt > E2.CodeEt}((\pi_{[E1.CodeEt, E1.Nom, E1.Promotion, E1.Commune]} ETUDIANT)\ E1 \bowtie (\pi_{[E2.CodeEt, E2.Nom, E2.Promotion, E2.Commune]} ETUDIANT\ E2)))$

2.4 Schématiser l'arbre algébrique correspondant. **(0,5 Pt)**



2.5 Présenter le plan d'exécution optimal. (**0,5 Pt)**

R1← Projection de la table Etudiant sur Nom, Promotion, Commune, CodeEt

R2← Projection de la table Etudiant sur Nom, Promotion, Commune, CodeEt

R3← Joindre R1 avec R2

R4← Sélection de la table R3 les lignes dont E1.Promotion=E2.Promotion et E1.Commune=E2.Commune et E1.CodeEt > E2.CodeEt

R5← Projection de R4 sur Nom Etudiant1 et Nom Etudiant2