Université A/Mira de Béjaia
Département d'informatique
Examen ABDD - L3
Date: 10/06/2017

Nom: _____
Prénom: _____
Groupe: _____

Exercice 1 (6 points):

1. (0.25 points) Dans quelle clause SQL est utilisé le mot clé LIKE ?

   WHERE

2. (0.5 points) Est-il possible de faire un JOIN d'une table avec elle-même ?

   Oui

3. (0.5 points) Est-il possible d'interroger plusieurs tables à la fois, avec une seule requête SQL ?

   Oui

4. (0.5 points) Comment s'appelle l'opération de liaison de données permettant d'interroger plusieurs tables à la fois ?

   Jointure

5. (0.5 points) En SQL, qu'est-ce qu'on utilise pour écrire un script de contrôle de BDD avec la programmation événementielle?

   Déclencheur

6. (0.25 X 2 points) Donner deux exemples de privilèges système ?

   CREATE USER, CREATE TABLE, DROP, BACKUP, ...

7. (0.25 X 2 points) Donner deux exemples de privilèges Objet ?

   UPDATE, INSERT, DELETE, SELECT, ...

8. (0.5 points) Comment appelle-t-on un regroupement de privilèges ?

   Rôle

9. Donner une brève description des commandes SQL suivantes :
   (a) (0.25 points) **ALTER**: **Modification d'un objet de base de données existant**, comme une table.
   (b) (0.25 points) **DROP**: **Suppression de toute une table, une vue ou un autre objet dans la base de données**.
   (c) (0.25 points) **REVOKE**: **Enlever les privilèges accordés aux utilisateurs**.

10. (0.5 points) Que va-t-il se passer si on n'ajoute pas la clause WHERE dans une requête de suppression ?

    Toutes les lignes de la table seront supprimées

11. (0.5 points) Quel est l'intérêt de l'utilisation des transactions ?

    Assurer le passage d'un état cohérent et correct vers un autre **état cohérent et correct** suite à l'exécution de plusieurs requêtes SQL

12. (0.5 points) Quel est l'objectif de l'optimisation des requêtes ?

    Trouver un plan d'exécution permettant **d'améliorer le temps de traitement de la requête**.

Exercice 2 (14 points): Soit le schéma relationnel suivant:

- ***Artiste*** (numArtiste, nomArtiste, prenomArtiste, ville)
- ***Chanson*** (numChanson, titre, genre, dateSortie, numArtiste)

Avec: *numArtiste, numChanson* sont de type **entier**. *nomArtiste, prenomArtiste, ville, titre* et *genre* de type chaine de caractère et *dateSortie* de type Date

1. (0.25 points) Ecrire une requête SQL pour la création de la base de données *ArtisteDB*

   - CREATE DATABASE ArtisteDB;

2. ((0.5 artiste - 01 chanson) points) Ecrire deux requêtes SQL pour la création des deux tables *Artiste* et *Chanson*.

Artiste	Chanson
CREATE TABLE Artiste(numArtiste Integer NOT NULL, nomArtiste VARCHAR(30), prenomArtiste VARCHAR(30), ville VARCHAR(30),);	CREATE TABLE Chanson(numChanson Integer, titre VARCHAR(30), genre VARCHAR(30), dateSortie date, numArtiste Integer NOT NULL, CONSTRAINT pk_Chanson PRIMARY KEY (numChanson));

3. (0.5 points) Ajouter la contrainte d'intégrité de clé primaire *numArtiste* sur la table *Artiste*.

   - ALTER TABLE Artiste ADD CONSTRAINT pk_Artiste PRIMARY KEY(numArtiste);

4. (0.5 points) Ajouter la contrainte d'intégrité de clé étrangère *numArtiste* sur la table *Chanson* qui fait référence au champ *numArtiste* de la table *Artiste*

   - ALTER TABLE Chanson ADD CONSTRAINT fk_numArtiste FOREIGN KEY(numArtiste) REFERENCES Artiste(numArtiste);

5. (0.5 points) Ajouter une colonne *style* (de type chaine de caractères), dans la table *Artiste* avec 'NoStyle' comme valeur par défaut.

   - ALTER TABLE Artiste ADD style VARCHAR(20) DEFAULT 'NoStyle';

6. (0.25 points) Insérer un artiste avec les attributs (*numArtiste*=1, *nomArtiste*='NomArt1', *prenomArtiste*='Pre *style*='Dance').

   - INSERT INTO Artiste (numArtiste, nomArtiste, prenomArtiste, style) VALUES (1, 'NomArt1', 'PreArt1', 'Dance');

7. (0.25 points) Mettre à jour *nomArtiste*='Spears', *prenomArtiste*= 'Britney' et *ville*='Mississippi' dans la table *Artiste* pour les lignes ayant *numArtiste*=1.

   - UPDATE Artiste SET nomArtiste = 'Spears', prenomArtiste = 'Britney', ville='Mississippi' WHERE numArtiste = 1;

8. (0.5 points) Sélectionner le nombre de chansons par artiste.

   - SELECT COUNT(numChanson) AS NbrChanson FROM Chanson GROUP BY (numArtiste);

9. (0.5 points) Créer un index sur la colonne *ville* de la table *Artiste*.

   - CREATE INDEX indVille ON Artiste(ville);

10. (0.25 points) Créer un utilisateur *consulViewUser* et un utilisateur *manipUser*

    - CREATE USER consulViewUser;

    - CREATE USER manipUser;

11. (1 point) Créer une vue *artChaView* permettant de consulter les champs: *nom*, *ville*, *titre* et *dateSortie*.

    - CREATE VIEW artChaView(nomArtiste, ville, titre, dateSortie)

    AS SELECT nomArtiste, ville, titre, dateSortie

    FROM Artiste, Chanson

    Where Artiste.numArtiste = Chanson.numArtiste;

12. (0.5 points) Attributer le privilège SELECT sur la vue *artChaView* à l'utilisateur *consulViewUser*.

    - GRANT SELECT ON artChaView TO consulViewUser;

13. (0.25 X 2 points) Attribuer les privilèges, insertion et mise à jour, sur les deux tables *Artiste* et *Chanson* pour l'utilisateur *manipUser*.

    - GRANT INSERT,UPDATE ON Artiste TO manipUser;

    - GRANT INSERT,UPDATE ON Chanson TO manipUser;

14. (0.25 points) Créer un rôle *adminArtiste*.

    - CREATE ROLE adminArtiste;

15. (0.5 points) Attribuer tous les privilèges sur la table *Artiste* pour le rôle *adminArtiste*.

    - GRANT ALL ON Artiste to adminArtiste;

16. ((-0.25 à chaque erreur) 02 points) Supposant qu'il existe initialement 10 tuples dans la table *Artiste* (dernier tuple ayant *numArtiste*=10). Ecrire une transaction contenant dans l'ordre: un BEGIN TRAN, 2 insertions dans *Artiste* (de votre choix), un COMMIT, une insertion d'un artiste, une mise à jour sur la ville de l'artiste *numArtiste*=1, une suppression de l'artiste *numArtiste*=7, un BEGIN TRAN, une sélection complète sur la table *Artiste*, un point d'arrêt P1, une mise à jour de la ville de l'artiste *numArtiste*=3, une insertion d'un artiste (de votre choix), une sélection complète sur la table *Artiste*, un ROLLBACK P1, puis une sélection complète sur la table *Artiste*.

```
BEGIN TRAN tranQuestion15

INSERT INTO Artiste (numArtiste, nomArtiste, prenomArtiste, style) VALUES (11, 'NomArt11', 'PreArt11', 'Dance11');

INSERT INTO Artiste (numArtiste, nomArtiste, prenomArtiste, style) VALUES (12, 'NomArt12', 'PreArt12', 'Dance12');

COMMIT;

INSERT INTO Artiste (numArtiste, nomArtiste, prenomArtiste, style) VALUES (13, 'NomArt13', 'PreArt13', 'Dance13');

UPDATE Artiste SET ville = 'Bougie' WHERE numArtiste = 1;

DELETE FROM Artiste WHERE numArtiste = 11;

BEGIN TRAN tranQuestion16

SELECT count(numArtiste) FROM Artiste;

SAVE TRAN Point;

UPDATE Artiste SET ville = 'Bougie' WHERE numArtiste = 3;

INSERT INTO Artiste (numArtiste, nomArtiste, prenomArtiste, style) VALUES (14, 'NomArt14', 'PreArt14', 'Dance14');

SELECT count(numArtiste) FROM Artiste;

ROLLBACK TRAN Point;

SELECT count(numArtiste) FROM Artiste;
```

(a) (0.25 X 3 points) Donner le nombre de tuples dans la table *Artiste* à chaque sélection?
- 1: 12 tuples, 2: 13 tuples et 3: 12 tuples

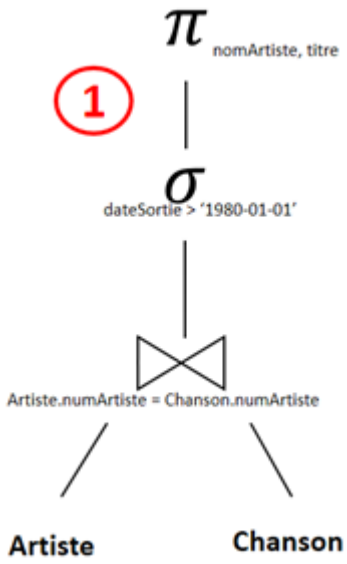
(b) (0.25 points) Donner le nombre de mises à jour validées
- une mise à jour validée

17. Soit la requête suivante: SELECT nomArtiste, titre FROM Artiste, Chanson WHERE Artiste.numArtiste = Chanson.numArtiste AND dateSortie > '1980-01-01'

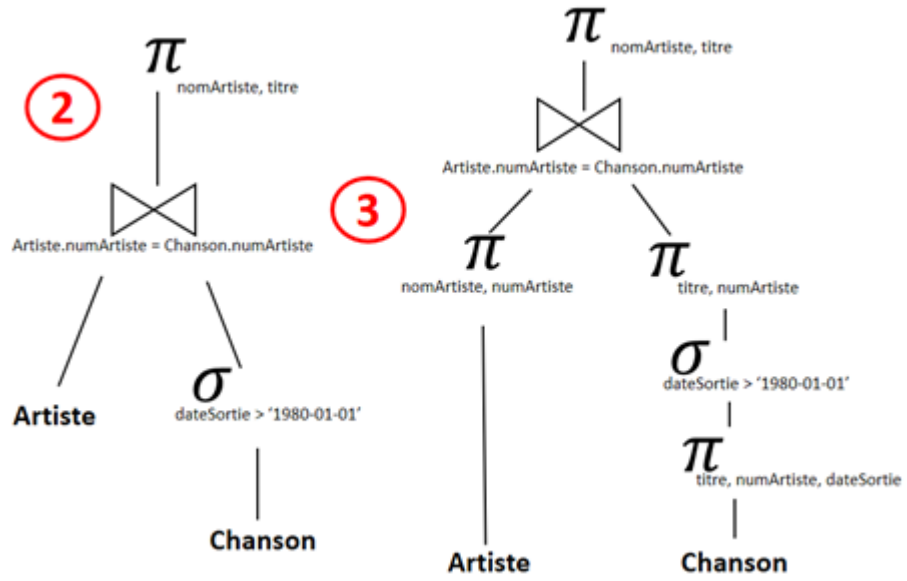
(a) (0.5 points) Donner l'expression algébrique.

$$\pi_{nomArtiste,titre}\left(\sigma_{dateSortie>1980-01-01}(Artiste \bowtie_{A.numArtiste=C.numArtiste} Chanson)\right)$$

(b) (0.5 points) Donner l'arbre algébrique



(c) ((0.25 si plan 2, 0.75 si plan 3) points) Donner un plan d'exécution optimisé.



(d) Supposant que la table Artiste comporte 20 tuples et la table Chanson 300 tuples et que 80% des artistes ont des chansons et 50% des chansons sont après '1980-01-01':
– Calculer le coût (en Entrée/Sortie) des opérateurs algébriques des deux plans (initial et optimisé)

A) **Plan initial** (0.25 X 2 pour plan initial) points:

- **Jointure**: 20E + (20*300)E + (300)S = 6320 E/S
- **Sélection** : 300E + (300*0.5)S = 450 E/S

B) **Plan Optimisé** (0.5 X 2 pour plan optimisé) points:

- **Sélection** : 300E + (300*0.5)S = 450 E/S
- **Jointure**: 20E + (20*150)E + 150S = 3170 E/S