

**INF3710 –Fichiers et Bases de données**

**Hiver 2019**

**TP No. 4**

**Groupe 4**

1340155 – Alassane Maiga

1862313 – Mohamed Esseddik BENYAHIA

**Soumis à : Manel Grinchi.**

**24 Mars 2019**

**Partie 2 – Algèbre relationnelle et SQL**

**1.1**

1. Retrouver les détails de tous les spectacles en 2010

annee = 2010Spectacle

1. Retrouver le détail de tous les danseurs qui ne sont pas dans la vingtaine

(age > 29) v (age < 20) Danseur

1. Retrouver le nom de tous les directeurs artistiques Canadiens

nom (nationalite = Canadien (DirecteurArtistique))

1. Retrouver le nom de chaque danseur ainsi que les titres des Spectacles dans lesquels il/elle s’est produit

nom, titre (Danseur ⨝ Performance ⨝ Spectacle)

1. Trouver les noms de tous les danseurs qui ont dansé le rôle du ‘cygne’ ainsi que l’année du spectacle

nom, annee (Danseur ⨝ (role = ‘cygne’ (Performance)) ⨝ Spectacle)

1. Retrouver toutes les informations des danseurs du Spectacle ‘Opus Cactus’ sans opération non nécessaire (indice : vous ne pouvez pas utiliser uniquement un join)

OpusCDanseur ⨝ Performance ⨝ (titre = ‘Opus Cactus’ (Spectacle))

DanseurId, nom, nationalite, age (OpusC)

1. Retrouver les titres de tous les spectacles dans lesquels les danseurs Philippe et Kate ont dansé ensemble

Spect\_Philippe  **(**(nom = ‘Philippe’ (Danseur)) ⨝ Performance ⨝ Spectacle**)**

Spect\_Kate  **(**(nom = ‘Kate’ (Danseur)) ⨝ Performance ⨝ Spectacle**)**

Réponse  titre (Spectacl\_Philippe ⨝ Spectacl\_Philippe.SpectacleId = Spectacl\_Kate.SpectacleId Spectacl\_Kate)

**1.2**

1. Quel est l’âge moyen des danseurs ? Stockez-le dans une colonne nommée AgeMoyen.

**Algèbre:**

R (AgeMoyen) AVG age (Danseur)

**SQL :**

SELECT AVG(age) AS AgeMoyen FROM Danseur ;

1. Quels danseurs (Nom) ont dansé dans au moins un spectacle où la danseuse Lucie Tremblay n’a pas dansé ?

**Algèbre:**

Spectacle\_Lucie  nom = ‘Lucie Tremblay’ (Danseur)⨝ Performances

Spectacle\_Sans\_Lucie SpectacleId (Performance)) – (SpectacleId (X))

Réponse  nom (Spectacle\_Sans\_Lucie ⨝ Danseur)

**SQL :**

SELECT nom FROM Danseur WHERE DanseurId IN

(SELECT DanseurId FROM Performance WHERE SpectacleId NOT IN

(SELECT SpectacleId FROM Performance p, Danseur d WHERE nom = “Lucie Tremblay” AND p.DanseurId = d.DanseurId));

1. Quel est le nombre de spectacles du danseur dont l’id = 1 ? Stockez le résultat dans une colonne nommée nbSpectacle.

**Algèbre:**

R(nbSpectacle)  COUNT SpectacleId (DanseurId = 1 (Performances))

**SQL :**

SELECT COUNT(\*) AS nbSpectacle FROM Performance WHERE DanseurId = 1;

1. Affichez une liste des danseurs ainsi que les spectacles (ID) qui leur sont associés s’ils existent, sinon affichez null. L’attribut en commun ne doit pas être répété.

**Algèbre:**

Danseur) ⟕ (DanseurId, SpectacleId (Performance))

**SQL :**

SELECT d.\*, p.SpectacleId FROM Danseur d NATURAL LEFT OUTER JOIN Performance p;

1. Combien de spectacles existent par catégorie ? Stockez le résultat en donnant un nom à la ou les colonnes correspondantes de la relation résultat.

**Algèbre:**

R (Catégorie, nbSpectacle)categorie  COUNT SpectacleId (Spectacle)

**SQL :**

SELECT categorie, COUNT(\*) AS nbSpectacle FROM Spectacle GROUP BY categorie;

1. Quels danseurs (affichez leurs détails) n’ont participé à aucun spectacle

**Algèbre:**

DanseurId Danseur) – DanseurId (Performance)) ⨝ Danseur

**SQL :**

SELECT \* FROM Danseur WHERE DanseurID NOT IN (SELECT DanseurID FROM Performance);

**Partie 3 – Transactions**

**1.**

1. Que se passe-t-il quand vous exécutez ces deux transactions concurrentes ? Quel est le problème ?

Ici, le problème est que la transaction A est complètement ignorée. Seule la valeur de la transaction B est prise en compte à la fin. En effet, la transaction A retire 200$ du compte et la transaction b retire 500$. On s'attendrait donc que le résultat final soit une réduction de 700$ du compte.

Mais on voit qu'on a juste le retrait de 500 de la transaction B dans l'état final. Ceci est dû au fait que la mise à jour du compte par la transaction B s'est faite sur la valeur lue précédemment par la transaction B. Une valeur qui avait été lue avant que la transaction A n'ait validé sa transaction. Ainsi, lorsque la transaction A écrit sur la table et valide, la valeur se fait écrasée par la transaction B.

|  |  |
| --- | --- |
| **Transaction A: Elle commence et termine avant la transaction B** | **Transaction B: lorsqu’elle valide, elle écrase la valeur du compte de la transaction A** |

1. Comment pourrions-nous nous assurer que les résultats soient cohérents ?

Pour éviter le problème, la transaction B doit attendre que la transaction A écrit la valeur sur le compte avant que la transaction B la lise. On doit donc verrouiller le tuple que la transaction A mettra à jour. On peut le faire avec un SELECT … FOR UPDATE. Ainsi la transaction B ne lira pas tout de suite la valeur du compte et attendra la fin de la transaction A avant de le faire.

|  |  |
| --- | --- |
| **Transaction A: Elle commence et termine avant la transaction B** | **Transaction B: lorsqu’elle écrit, elle utilise, comme attendu, la valeur du compte de la transaction A** |

**2.**

1. Quel problème constatez-vous avec READ COMMITED?

Quand la tranaction A utilise l'isolation READ COMMITTED, si la transaction B fait un commit pendant que A est encore en cours, la transaction A lit des valeurs différentes lorsqu’elle fait des SELECT sur les tuples que B écrit et valide. En effet: le deuxième SELECT utilise la nouvelle valeur des montants dans les comptes comme écrit par B.

|  |  |
| --- | --- |
| **Transaction A: La valeur du select dépend des valeurs qui ont été validée concouramment** | **Transaction B: Transfert entre les deux comptes** |

1. Que se passe-t-il si changez le niveau d’isolation de la transaction A à REPEATABLE READ?

Avec Repeatable READ par contre, lorsque la transaction B fait un commit, la transaction A utilise encore les anciennes valeurs de Accounts: celle qui ont été validées avant le début de la transaction A.

|  |  |
| --- | --- |
| **Transaction A: elle lit toujours les mêmes valeurs sur les même tuples** | **Transaction B: transfert entre les deux comptes** |

1. Quel problème constatez-vous avec REPEATABLE READ READ ONLY?

Similairement lorsqu'on utilise l'isolation REPEATABLE READ en mode Lecture seule, La transaction A voit les valeurs de la table qui ont été validée avant que la transaction commence. Donc quand la transaction B valide sa transaction, ceci est toujours invisible à la transaction A. Le mode “read only” n'affecte que la transaction A, l'empéchant de modifier les tables. La transaction B peut encore changer les données des tables.

3. Nous voyons dans les figures qui suivent une situation d’interblocage. Le système abandonne la transaction A pour débloquer les transactions.

|  |  |
| --- | --- |
| **Transaction A: Détection de l’interblocage** | **Transaction B: la dernière mise à jour cause une situation d’interblocage** |