Examen Machine 3 - Bases de Données

CS54 : Computer Science 54
Première année

Lundi 22 Novembre 2021
Durée : 4 heures

Propos liminaires

Vous disposez de 4 heures pour réaliser tout ou partie de l'examen. Prenez le temps de lire les questions, de les comprendre, de réfléchir à des solutions. Il existe de multiples manières de résoudre les problèmes posés.

Assurez-vous de bien déposer vos productions sur le dépôt git qui vous a été attribué pour l'examen, les évaluations étant automatiques. Ne copiez pas, ne trichez pas. Les conséquences (conseil de discipline, exclusion, …) sont trop lourdes pour s'y risquer.

Préparation de l'environnement de travail

Récupérez (par clonage) une version locale du projet git qui vous a été affecté pour cet examen.

Au besoin, référez vous à la page Arche de CS54 pour les bases de Git et Gitlab :

https://arche.univ-lorraine.fr/mod/page/view.php?id=1252876

Connexion au serveur de base de données Postgres SQL :

Veuillez lancer **pgAdmin** et créez une instance du serveur avec l'option "*Create Server* ".
Choisissez un nom de connexion et utilisez *localhost* comme adresse du serveur (port *5432*, nom d'utilisateur : *postgres* et mot de passe : *votre mot de passe (défini lors de l'installation de postgressql)*).
Pour interroger le schéma de la base de données, il suffit de sélectionner le sous menu *postgres* du menu *Databases* de votre serveur et puis de lancer l'outil "*Query Tool*" (Menu *Tools* de l'interface principale de ***pgAdmin***).
Saisissez vos requêtes dans la fenêtre *Query editor* de l'outil *Query tool* !
Vous pouvez aussi utiliser Postgres SQL en ligne de commande.

Supports de cours : Les supports de cours sont disponibles sur Arche.
Documentation de PostgresSQL : https://www.postgresql.org/docs/
Tutorial avec des éléments de syntaxe PL/PGSQL :
https://www.postgresqltutorial.com/postgresql-plpgsql/

Examen Machine 3 - Bases de Données

CS54 : Computer Science 54

Première année

Sujet d'examen et modalités de rendu

- Le sujet d'examen est disponible sur Arche.
- Vous sauvegarderez au fur et à mesure vos **requêtes** dans un fichier texte (ou sql). Le nom du fichier texte doit respecter le format suivant : **NOM_PENOM_TP_BD.txt** ou **NOM_PENOM_TP_BD.sql.** Pour chaque question, mettez le numéro de la question sou forme de commentaire (sur une seule ligne avec - - ou sur plusieurs lignes avec /* … */).
- Les projets *git* sont "clonable" à partir de 14h. Il faudra déposer votre travail sur git avant 18h.

C'est parti !

On souhaite créer les schémas des relations suivantes :

CLIENT (NOMPRENOM, DATENAISS, ADRESSE, PROFESSION)

TYPECOMPTE (NOM, CATEGORIE, SOLDEMIN, INTERET)

COMPTE (NOCOMPTE, #TYPEC, #PROPRIETAIRE, SOLDE)

TRANSACTIONBANCAIRE (#NOCOMPTE, DATET, TYPETRANSAC, CREDIT, DEBIT)

Liste des contraintes attachées au schéma de la base de données :

- Les clés primaires sont en gras et soulignées. On suppose que le couple *(NOM, PRÉNOM)* d'un client est unique.
- Les clés étrangères sont identifiées par **#**. Le propriétaire d'un compte est un CLIENT. Le type d'un compte et un nom de TYPECOMPTE.
- 4 valeurs possibles de *TYPECOMPTE.NOM* : compte courant étudiant, compte courant, compte sur livret, compte actions.
- 3 valeurs possibles de *catégorie* : courant, épargne, titres.
- 4 valeurs possibles de *TYPETRANSAC* : débit espèces, débit chèque, crédit espèces, crédit chèque.
- Une transaction correspond soit à un crédit ou dépôt d'argent (dans ce cas, *DEBIT* = 0), soit à un débit ou retrait d'argent (dans ce cas, *CREDIT* = 0).
- Par défaut, la date et l'heure d'une transaction est la date et l'heure courante, renvoyée par la fonction *CURRENT_DATE*.
- Les attributs *SOLDEMIN, SOLDE, CREDIT, DEBIT* sont de type réel (20 chiffres dont 2 décimales)
- L'attribut *INTERET* est un pourcentage compris entre 0 et 100 (un intérêt de 2.5 signifie que le compte rapport 2.5% par an)

Partie I : SQL (25 points)

1. Créer les schémas de ces relations.
2. Afficher les noms de colonnes de la table **CLIENT** en interrogeant le schéma d'information (*information_schema*), un ensemble de vues contenant des informations sur les objets définis dans la base de données courante : https://docs.postgresql.fr/14/information-schema.html
3. Retrouver quelques informations (nom de la contrainte, expression de vérification de la contrainte) sur les contraintes définies dans la question 1 pour les tables **COMPTE** et **TRANSACTIONBANCAIRE** en interrogeant le schéma d'information : https://docs.postgresql.fr/14/information-schema.html
4. Insérer les *tuples* suivants dans la base de données. Pour cela, vous devez exécuter les requêtes fournies dans le fichier *data.sql* disponible dans le dossier "**Examen TP BD**" sur **Arche**.

CLIENT

NOMPRENOM	*DATENAISS*	*ADRESSE*	*PROFESSION*
Olivier Durand	15/02/1989	21 Avenue de General Leclerc 54000 Nancy	Etudiant
Marine Vogt	12/12/1990	21 Avenue Jean Jaures 54500 Vandoeuvre les Nancy	Ingénieur d'étude
…	…	…	…

TYPECOMPTE

NOM	*CATÉGORIE*	*SOLDEMIN*	*INTÉRÊT*
compte courant	courant	-500	0.5
compte courant étudiant	courant	-200	0.5
compte sur livret	épargne	-1000	2
…	…	…	…

COMPTE

NOCOMPTE	*TYPEC*	*PROPRIÉTAIRE*	*SOLDE*

E-120-256	compte courant étudiant	Olivier Durand	-100
C-005-569	compte sur livret	Marine Vogt	5260
…	…	…	…

TRANSACTIONBANCAIRE

noCOMPTE	*dateT*	*typeTRANSAC*	*CRÉDIT*	*DÉBIT*
L95129	03/04/2018	crédit espèces	100	0
C-005-569	03/04/2018	crédit espèces	150	0
…	…	…	…	…

5. Supprimer de la table **COMPTE** les comptes des enseignants qui habitent à Nancy.
6. Afficher les transactions qui concernent les comptes courants ou les transactions dont le numéro de compte commence par la lettre 'C' et contient au maximum 6 caractères.
7. Ajouter une contrainte à la table **TYPECOMPTE** pour que la valeur de l'attribut *INTERET* soit toujours supérieure à 0 et inférieure à 3.
8. Créer une vue **VueClients** contenant les informations des clients (nomPrenom, **age**, adresse, profession et noCompte) qui ont un ou plusieurs comptes courants. Pour calculer l'age, vous pouvez utiliser la fonction prédifinie age : https://www.postgresql.org/docs/8.4/functions-datetime.html
9. Afficher le contenu actuel de la **VueClients**, limité aux attributs nomPrenom, adresse et age. L'affichage devra se faire par ordre alphabétique sur le nom et le prénom.
10. Donner le nombre de clients pour chaque type de compte.
11. Afficher les clients qui ont au moins un compte de chaque type de compte.
12. Afficher les noms des clients dont le nom contient trois fois la lettre 'o' ou comportant exactement 15 caractères.
13. Afficher les noms et prénoms des clients qui n'ont aucun compte courant
14. Afficher les noms, prénoms, le type de compte et la date de transaction des clients qui n'ont pas fait une transaction bancaire depuis le 01/04/2021.
15. Afficher le nombre de clients dans les deux communes de Meurthe-et-Moselle (code postal 54000 et 54500).

16. On veut maintenant limiter la recherche précédente à la ville de Nancy (code postal 54000) ayant plus d'un client avec comme profession "Enseignant".
17. Afficher les noms, prénoms et adresses des clients de la ville de Nancy qui ont le même type de compte bancaire.
18. Afficher pour chaque client, la somme des montants de tous ses comptes.
19. Pour les clients qui ont un solde négatif, afficher la date, le montant et le type de leur dernière transaction bancaire.

Partie II : PL/PGSQL (15 points)

Exercice 1 (4 points) :

1. Écrire une procédure nommée **MajSolde()** qui permet de mettre à jour les soldes des comptes de tous les clients en consultant l'ensemble de transactions concernées.
2. Tester la procédure **MajSolde()**.
3. Transformer la procédure **MajSolde()** pour en faire une procédure **MajSolde_V2(nom)** qui prend en paramètre le nom du client et qui met à jour le solde du client.
4. Tester la procédure **MajSolde_V2('Marine Vogt')** pour le client "Marine Vogt".

Exercice 2 (2 points) :

1. Écrire une procédure nommée **FiltreClient(N)** qui permet d'afficher les informations (nomPrenom et adresse) sur les clients âgés de plus de N ans. N est un paramètre de la procédure.

   Contraintes :

   - Utiliser le typage dynamique pour définir les types des variables et les paramètres de la procédure.
   - Traiter correctement au moins l'exception correspondant à un résultat vide (pas de client âgé de plus de N ans). Vous devez lever une exception quand la procédure **FiltreClient** renvoie un résultat vide.
   - Pour calculer l'age, vous pouvez utiliser la fonction prédifinie age : https://www.postgresql.org/docs/8.4/functions-datetime.html

2. Tester la procédure **FiltreClient(25)** pour chercher les clients âgés de plus de 25 ans.

Exercice 3 (3 points) :

1. Écrire une procédure nommée **Infos_Clients()** permettant d'afficher tous les clients, et pour chaque client, la liste des comptes, et pour chacun de ces comptes, l'historique des opérations.
2. Tester la procédure **Infos_Clients().**

Exercice 4 (3 points) :

1. Écrire une procédure nommée **Ouvrir_Nouveau_Compte (NOMPRENOM, ID_COMPTE, TYPE_COMPTE)** permettant de créer pour un client donné en paramètre un nouveau compte bancaire selon son type. Ajouter des tests permettant de vérifier l'existence des comptes clients, type de compte, etc.
2. Tester la procédure **Ouvrir_Nouveau_Compte ('Vincent Mollard', 'E-194-202', 'compte courant étudiant')**

Exercice 5 (3 points) :

1. Écrire une procédure **CALCUL_DEPENSES(NOMPRENOM)** permettant de calculer pour un client donné en paramètre la somme des dépenses effectuées durant le mois courant (la période entre le 01 du mois courant et la date courante).
2. Tester la procédure **CALCUL_DEPENSES('Christophe Lemaire')**.