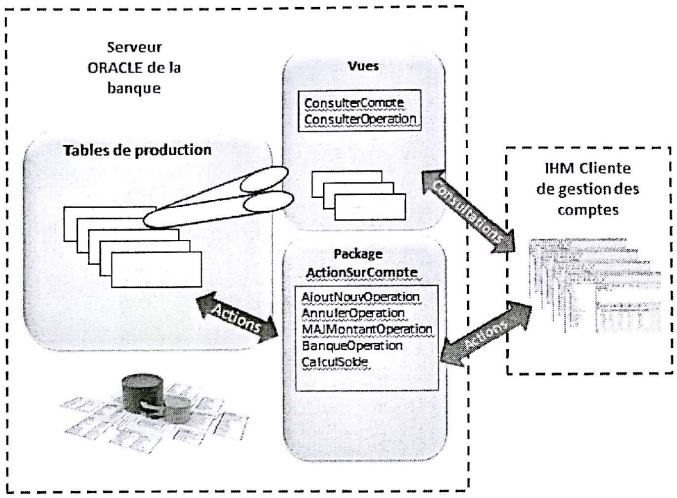
DST SGBD2

2015-2016
Jérôme FESSY

Tout document autorisé
Durée: 1h30

Le contexte de cet examen est l'application bancaire travaillée ces deux dernières semaines. Le schéma ci-dessous résume son architecture :



On donne le schéma relationnel utile des tables de production :

BANQUE *(IdBanque, LibelleBanque, VilleBanque)*
COMPTE*(IdCompte,LibelleCompte , IdBanque#, SoldeCompte)*
OPERATION *(IdOp,DateOp,MontantOperation, IdCompte#)*

Les clés primaires sont soulignées ; les clés étrangères précédées d'un #
La colonne SoldeCompte a été ajoutée

On considère que toutes les procédures, fonctions et les deux vues sont implantées et fonctionnent.

1. Ajout de procédures

On souhaite ajouter les deux procédures suivantes :

- ✓ ***TRANSFERTCOMPTE****(CptOrig INTEGER, CptDest INTEGER, Value NUMBER)*
  Cette procédure permet de réaliser un transfert d'argent entre comptes. Elle prend en paramètre l'identifiant du compte d'origine, l'identifiant du compte destination et le montant du transfert. Ce montant est toujours positif.

- ✓ ***GESTIONSOLDE()***
  Cette procédure sans paramètres permet, lorsqu'elle est appelée, de renseigner la colonne « SoldeCompte » de l'ensemble des comptes de la base. Cette colonne stocke le solde d'un compte à partir de la somme des opérations sur ce compte.

a) Donner le code PLSQL de ces deux procédures
b) Donner un exemple d'appel pour chacune d'elles.
c) Donner le code SQL ou PLSQL qui permet d'afficher le compte qui a le découvert le plus élevé.

2. Modification d'une procédure

On souhaite à présent modifier/compléter le code de la procédure « AjoutNouvOperation » pour que le solde d'un compte soit mis à jour directement lors de l'ajout d'une opération sur ce compte.

Proposer le code PLSQL complet de la procédure « AjoutNouvOperation ».

3. Analyse d'un code

Le code suivant vous est soumis :

```
CREATE OR REPLACE PROCEDURE P_A_ANALYSER
IS
    CURSOR C1 IS
    SELECT C.IdCompte, C.LibelleCompte, COUNT(*) AS NB
    FROM Compte C, Operation O WHERE C.IdCompte=O.IdCompte
    GROUP BY C.IdCompte, C.LibelleCompte;

    NbOperation INTEGER;
    Ratio NUMBER(10,2);

BEGIN

SELECT COUNT(*) INTO NbOperation FROM Operation;

FOR line IN C1 LOOP

Ratio := line.NB/NbOperation;

IF Ratio >= 0.5 THEN
    DBMS_OUTPUT.PUT_LINE('Compte ' || line.IdCompte|| ' : Elevé') ;
ELSIF Ratio >= 0.2 THEN
    DBMS_OUTPUT.PUT_LINE('Compte ' || line.IdCompte|| ' : Moyen') ;
ELSE
    DBMS_OUTPUT.PUT_LINE('Compte ' || line.IdCompte|| ' : Faible') ;
END IF;

END LOOP;
END;
/
```

Nb : La fonction PUT_LINE du package DBMS_OUTPUT permet d'afficher (chaine de caractères) un message sur la console.

a) Donner le résultat de l'exécution de P_A_ANALYSER sur le jeu de données suivant :

TABLE COMPTE

IdCompte	LibelleCompte
1	CC1
2	CC2
3	CC3

TABLE OPERATION

IdOperation	*IdCompte*	*MontantOperation*
1	2	-100
2	2	300
3	1	150
4	2	-50
5	1	400
6	3	250

b) On vous demande d'expliquer précisément ce que fait cette procédure.

4. Procédure d'administration

La procédure « DROP_ALL_INDEX » permet de supprimer tous les indexes de la base. Elle ne prend pas de paramètres. Les indexes sont des objets de la base référencés dans la table système « USER_INDEXES ». Cette table possède plusieurs colonnes dont la colonne « INDEX_NAME » qui donne le nom de l'index.

Il vous est demandé d'écrire le code PLSQL de la procédure « DROP_ALL_INDEX ».

NB :
Syntaxe SQL de suppression d'un index : « DROP INDEX <indexname> »

FIN