R3.07 - SQL dans un langage de programmation

TP3 - Banque (fin)

2022/2023 - A. Ridard

Objectifs

- Implémenter l'auto-incrémentation à l'aide d'un trigger et d'une séquence
- Modifier la structure d'une base de données suite à un changement de modélisation
- Analyser l'impact sur la gestion des contraintes et mettre à jour la programmation des triggers
- Automatiser une "opération" à l'aide de fonctions et de procédures
- Réaliser des tests

I. Auto-incrémentation des clés primaires et modification de la structure

Commençons par implémenter l'auto-incrémentation des clés primaires simples.

AUTO_INCREMENT disponible sous MySQL n'avait pas d'équivalent sous Oracle 11g.
Heureusement, GENERATED BY DEFAULT AS IDENTITY est maintenant disponible sous Oracle 12c.
L'inconvénient est que l'on ne contrôle pas la séquence sous-jacente qui, par exemple, ne peut pas être réinitialisée.

1. A l'aide du code ci-dessous, implémenter l'auto-incrémentation de la clé primaire **numAgence**.

```
DROP SEQUENCE seq_Agence;
CREATE SEQUENCE seq_Agence;

CREATE OR REPLACE TRIGGER trig_seq_Agence
BEFORE INSERT ON Agence
FOR EACH ROW
WHEN (NEW.numAgence IS NULL)
BEGIN
    SELECT seq_Agence.NEXTVAL INTO :NEW.numAgence
    FROM DUAL;
END;
/
```

2. Procéder de la même manière pour les autres clés primaires simples.

Nous allons maintenant changer la manière de stocker les directeurs des agences (cf. annexe pour l'analyse complète).

Changement de modélisation

Le directeur n'est plus modélisé par un booléen dans la table Agent mais par une clé étrangère dans la table Agence. Cette dernière engendre une double dépendance fonctionnelle entre les tables Agent et Agence qu'il convient de gérer lors de la création/destruction des tables, mais aussi lors de l'insertion/suppression de lignes dans ces tables (savez-vous comment ?)

3. **Sans toucher au script de création de tables**, modifier la structure des tables **Agent** et **Agence** en conséquence.

II. Mise à jour de la programmation des triggers

Une contrainte textuelle peut nécessiter plusieurs triggers, c'est le cas ici (cf. annexe pour plus de détails)

4. Pour gérer la contrainte « ***Un directeur travaille forcément dans l'agence qu'il dirige*** », écrire :
   (a) un trigger de ligne **trig_Agence_agDir** pour gérer la modification d'un directeur dans la table Agence.
   (b) un trigger de ligne **trig_Agent_agDir** pour gérer la modification de l'agence d'un directeur dans la table Agent.

5. Pour gérer la contrainte « ***Le directeur d'une agence est mieux payé que les agents de son agence*** », écrire :
   (a) un trigger de ligne **trig_Agence_salDir** pour gérer la modification d'un directeur dans la table Agence.
   (b) un trigger d'état **trig_Agent_salDir** pour gérer les deux modifications [a] dans la table Agent.

6. Tester les triggers à l'aide du script de test disponible sur Moodle.

a. pouvant remettre en question cette contrainte

III. Une demande de virement

Nous souhaitons enfin gérer une demande de virement (interne à la banque mais entre des clients éventuellement différents) :
- Si le compte émetteur est débiteur, le virement est refusé et un message d'erreur est envoyé.
- Sinon, le virement est réalisé et modifie la base de données en conséquence.

7. Écrire une fonction **fct_estDebiteur** définie par :
   - **Entrée** : le numéro de compte
   - **Sortie** : 1 si le compte est débiteur (solde négatif) et 0 sinon

8. Écrire une procédure **proc_ajoutOperation** définie par :
   - **Entrées** : le type d'opération, le montant, le client [a] et le compte
   - **Actions** : insère une ligne dans la table Opération et modifie le compte en conséquence

9. Programmer et tester la procédure de virement [b].

a. A quoi sert ce paramètre ?
b. **(Facultatif)** Si le montant dépasse 1000 euros, le virement est refusé et un message d'erreur est envoyé précisant le nom de l'agent qui gère le compte

Aucun travail n'est attendu concernant ces deux points, il s'agit simplement de remarquer la cohérence.

- Si le montant du virement dépasse le solde du compte émetteur, la contrainte textuelle (*) ne sera pas respectée, et donc le trigger du TP1 bloquera le virement.
- Si le compte émetteur n'apppartient pas au client émetteur, la contrainte textuelle (**) ne sera pas respectée, et donc le trigger du TP1 bloquera le virement.

Analyse du changement de modélisation

En violet, la modélisation du directeur du TP1 – en bleu, celle du TP3.

Agence (numAgence (1), telAgence, adAgence, sonDirecteur = @Agent(numAgent) (NN) (UQ))

Agent (numAgent (1), nomAgent (NN), prenomAgent (NN), salaire (NN), estDirecteur, sonAgence = @Agence(numAgence) (NN))

Client (numClient (1), nomClient (NN), prenomClient (NN), adClient, dateNaissClient, sonAgent = @Agent(numAgent) (NN))

Compte (numCompte (1), solde, typeCompte)

Operation (numOperation (1), dateOperation, typeOperation, montant, leClient = @Client(numClient) (NN),
leCompte = @Compte(numCompte) (NN))

Appartient (unCompte = @Compte(numCompte) (1), unClient = @Client(numClient) (1))

/!\ Il existe maintenant une double dépendance fonctionnelle entre les tables Agent et Agence

Contraintes n'apparaissant pas dans le SR mais gérées par le LDD :

- Aucun salaire ne doit être inférieur au Smic (1480 brut)
- Le type de compte est soit COURANT soit EPARGNE
- Le type d'opération est soit RETRAIT soit DEPOT
- Le montant d'une opération est toujours positif
- Par défaut, la date d'opération est la date du jour courant

Contraintes gérées par un trigger (de ligne) :

- Une augmentation de salaire ne doit pas dépasser 10% et une baisse 8%
- Un client ne peut pas être conseillé par un agent portant le même nom que lui
- Un client ne doit pas pouvoir effectuer un retrait dont le montant est supérieur au solde (*)
- Un client ne doit pouvoir retirer de l'argent que sur un compte qui lui appartient (**)
- Le directeur d'une agence est mieux payé que les agents de son agence
  - Modification d'un directeur dans la table Agence
- Un directeur travaille forcément dans l'agence qu'il dirige (toujours vérifiée avec l'ancienne modélisation)
  - Modification d'un directeur dans la table Agence
  - Modification de l'agence d'un directeur dans la table Agent

Contraintes gérées par un trigger (de table) :

- Une agence a forcément un et un seul directeur (gérée par le LDD avec la nouvelle modélisation)
- Le directeur d'une agence est mieux payé que les agents de son agence
  - Modification de l'agence d'un employé dans la table Agent
  - Modification du salaire d'un directeur ou d'un employé dans la table Agent

Contrainte gérée *a posteriori* par une vue :

- Tout compte appartient à au moins un client

Récupération des informations dérivables avec une vue :

- (numClient, ageClient, numAgence)