Alexis N3 Informatique

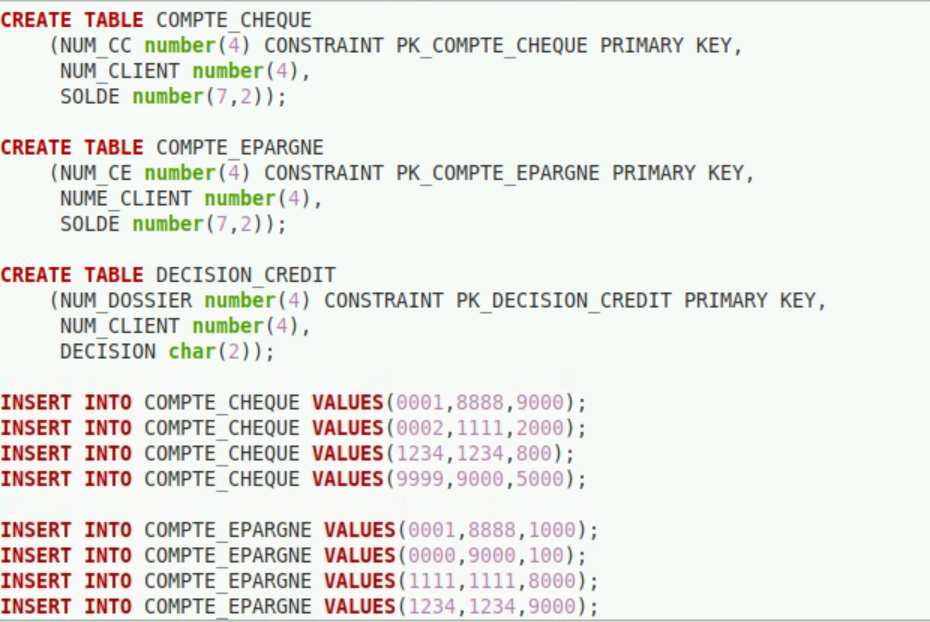
PLESSIER

11/01/2021

Implémentation des SGBD

Rendu TP2

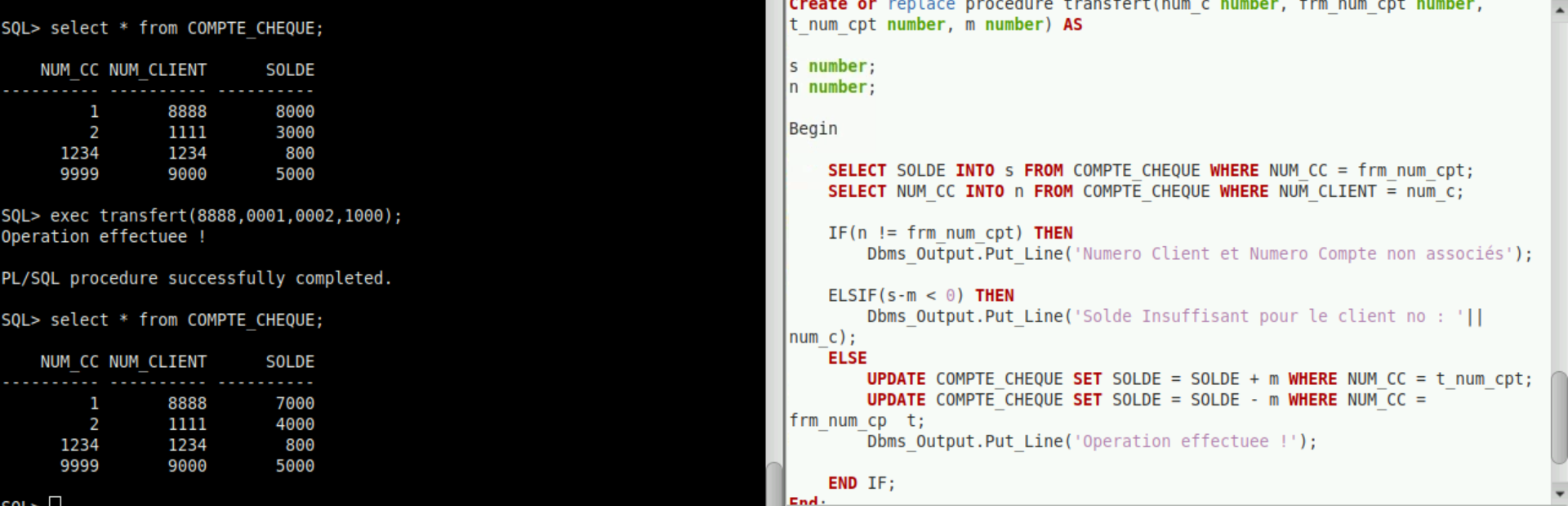
Exercice 1 :



Exercice 2 :

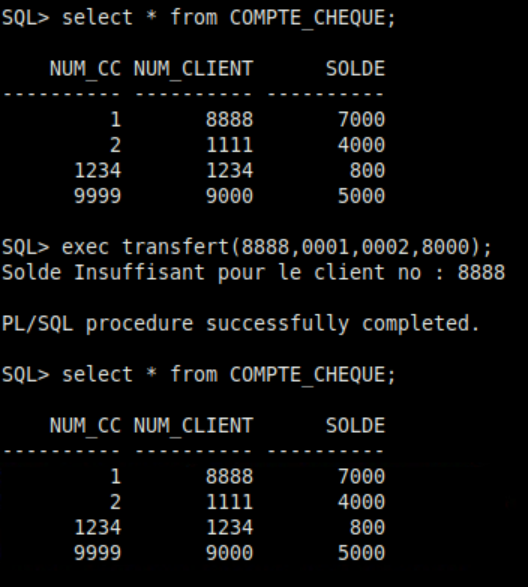
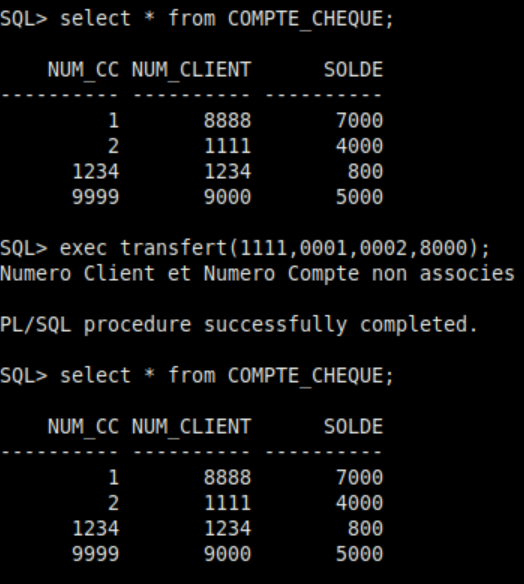
(a)

**Solde suffisant :**



Le client numéro 8888 a bien virer 1000 unités de son compte (0001) au compte 0002 du client numéro 1111.

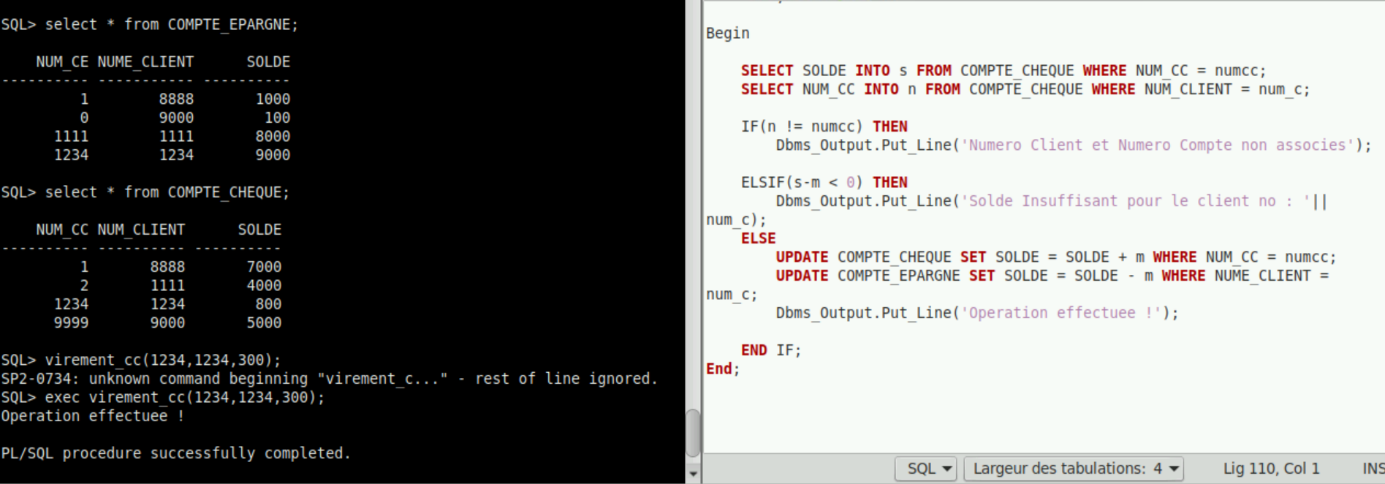
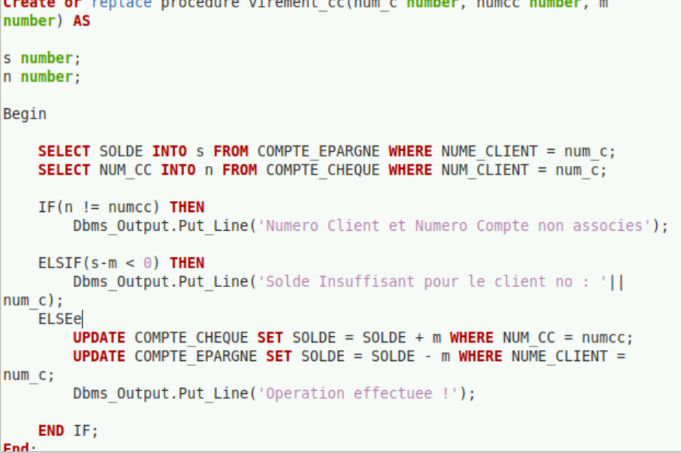
**Solde insuffisant : Erreur Numéro Client/Compte :**

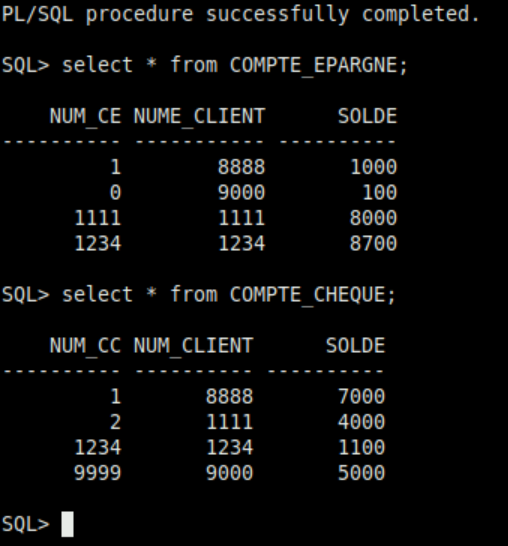
 

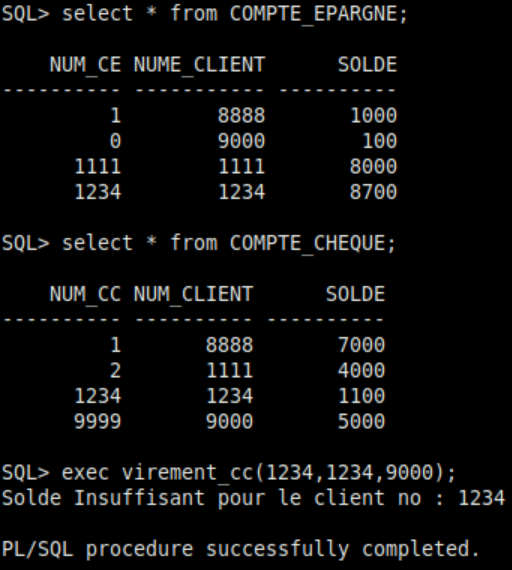
(b)

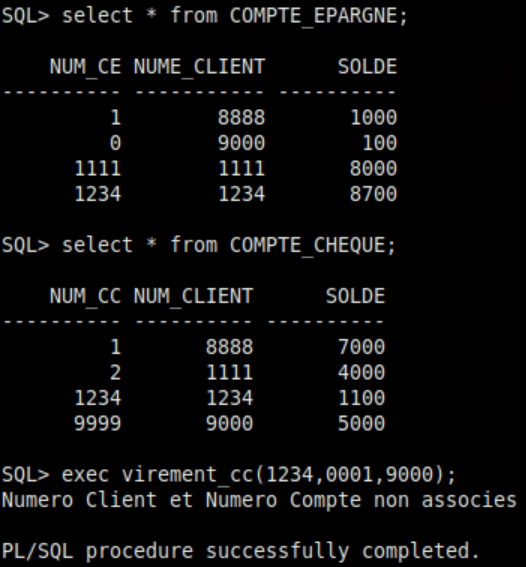
**Solde suffisant :**

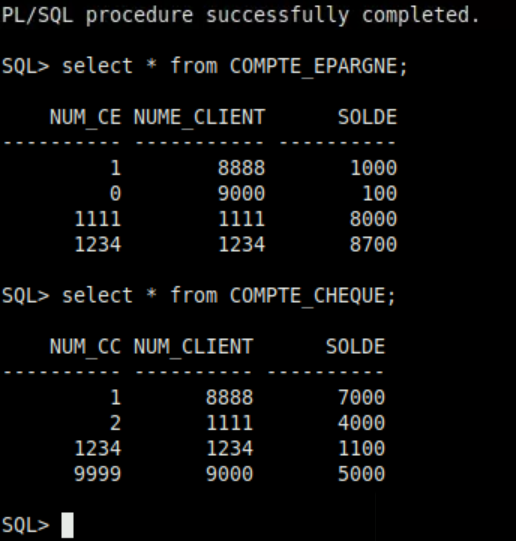
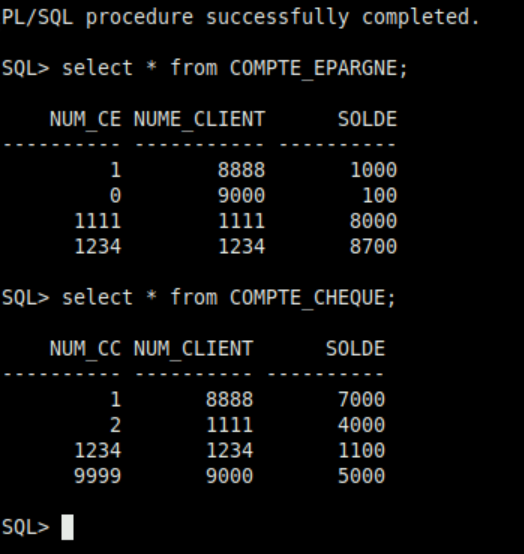
Le client numéro 1234 a bien viré 300 unités de son compte épargne à son compte chèque.





**Solde insuffisant : Erreur Numéro Client/Compte :**

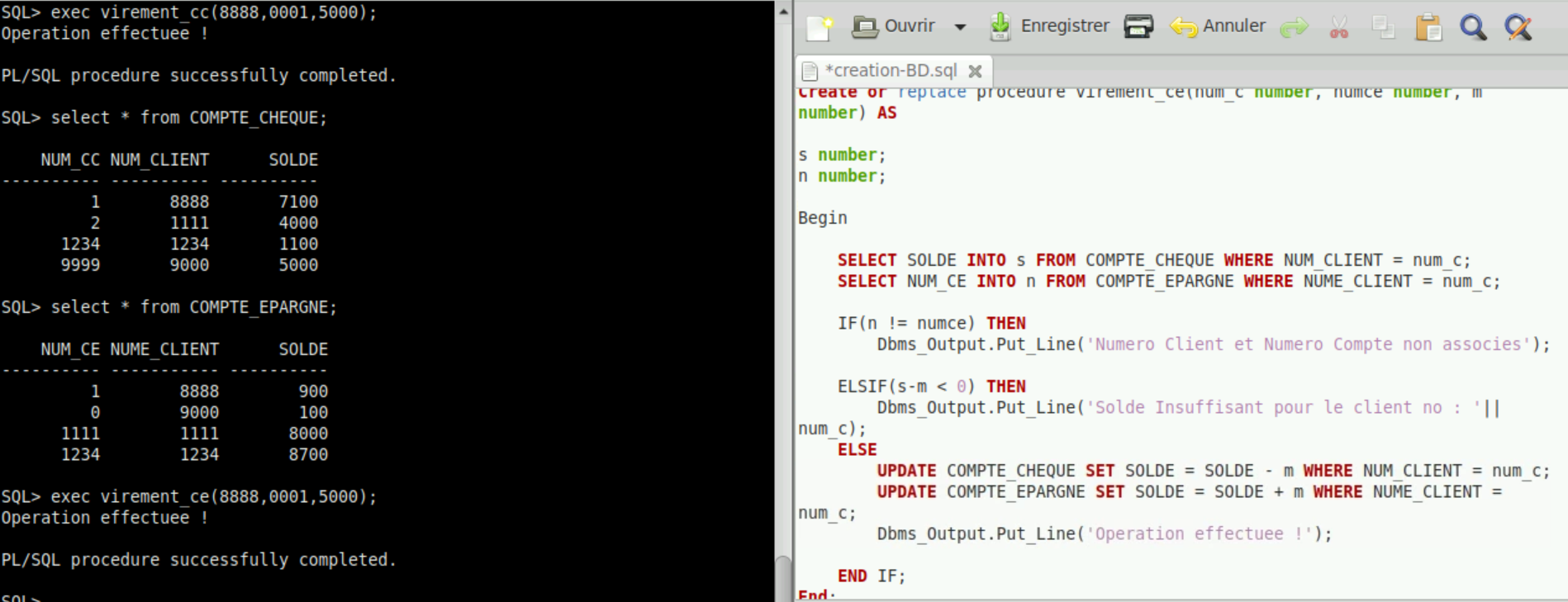
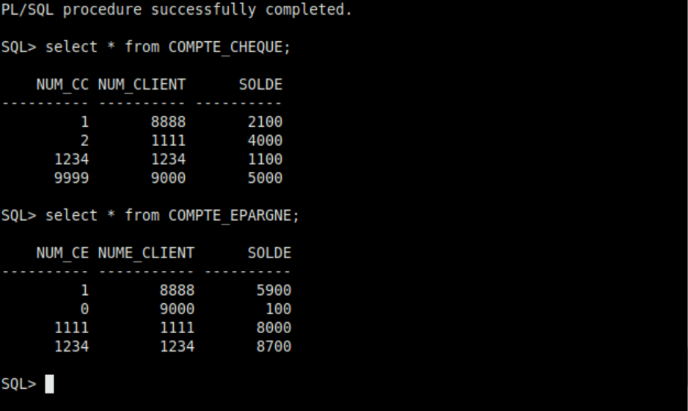




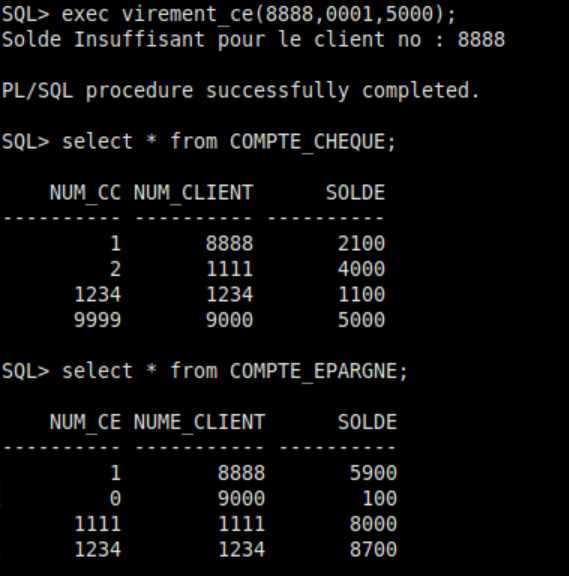
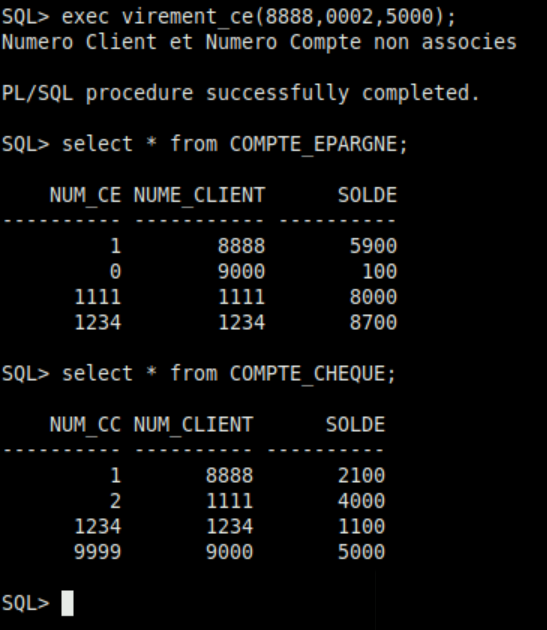
(c)

**Solde suffisant :**

Le client numéro 8888 a bien viré 5000 unités de son compte chèque à son compte épargne.

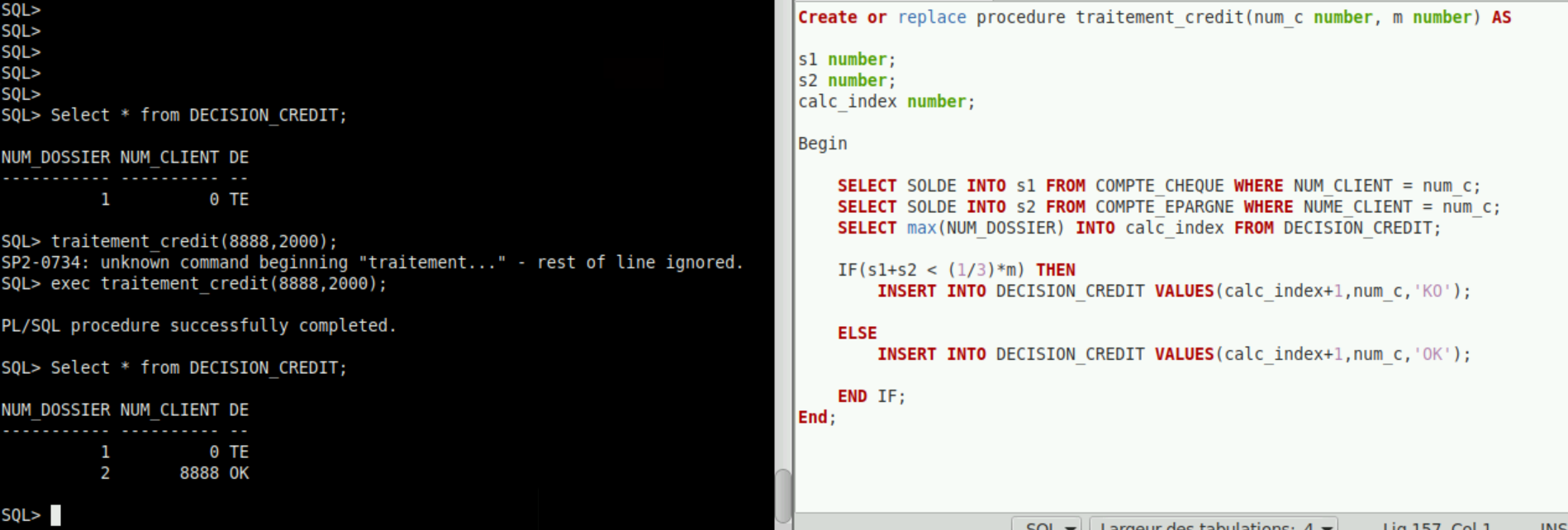


**Solde insuffisant : Erreur Numéro Client/Compte :**

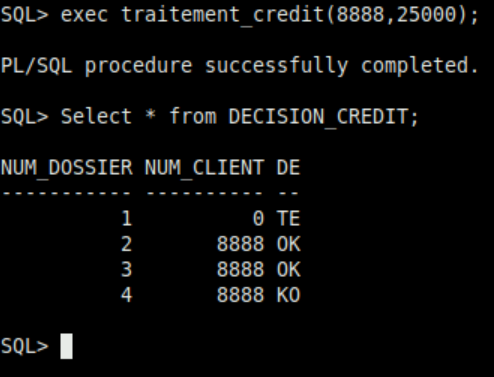
 

(d)

**Crédit accepter :**



**Crédit refuser :**



Exercice 3 :

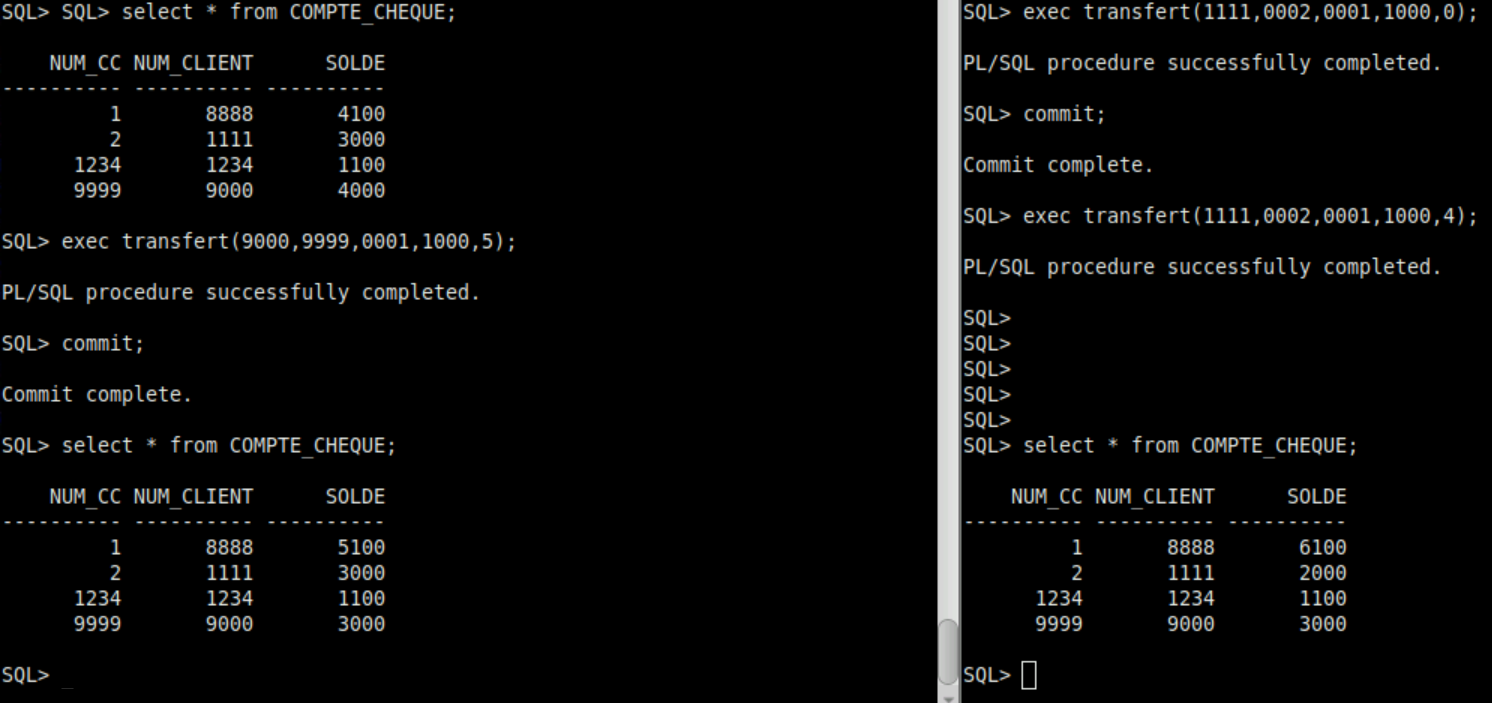
1. Ajout de *dbms\_lock.sleep(nb\_seconds IN NUMBER)* dans les procédures. Pour une question pratique, toutes les procédures se sont vu attribuer un paramètre en plus qui est « **sec number** ». Cette variable est ensuite appliquée à la fonction DBMS\_LOCK.sleep(sec) dans les procédures.

(b)

Transfert d’argent de deux personnes différentes sur un même compte :

Deux comptes envoient 1000 unités sur le compte numéro 0001 qui possède un solde de 4100. Sans problème de concurrence, le solde devrait être de 6100.

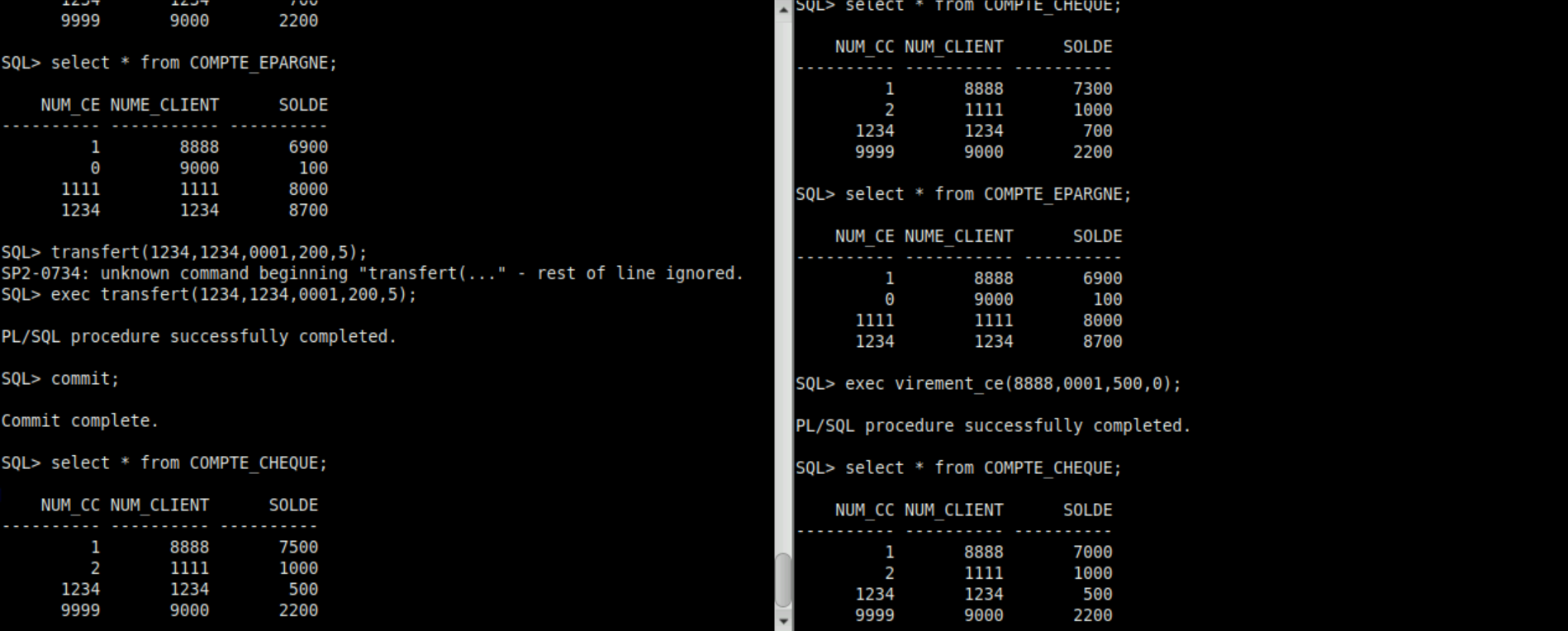
Session1 Session 2



On remarque que la session 1 possède une base de données erronée.

Transfert d’argent (d’un tiers) sur le compte chèque ET virement du compte chèque au compte épargne :

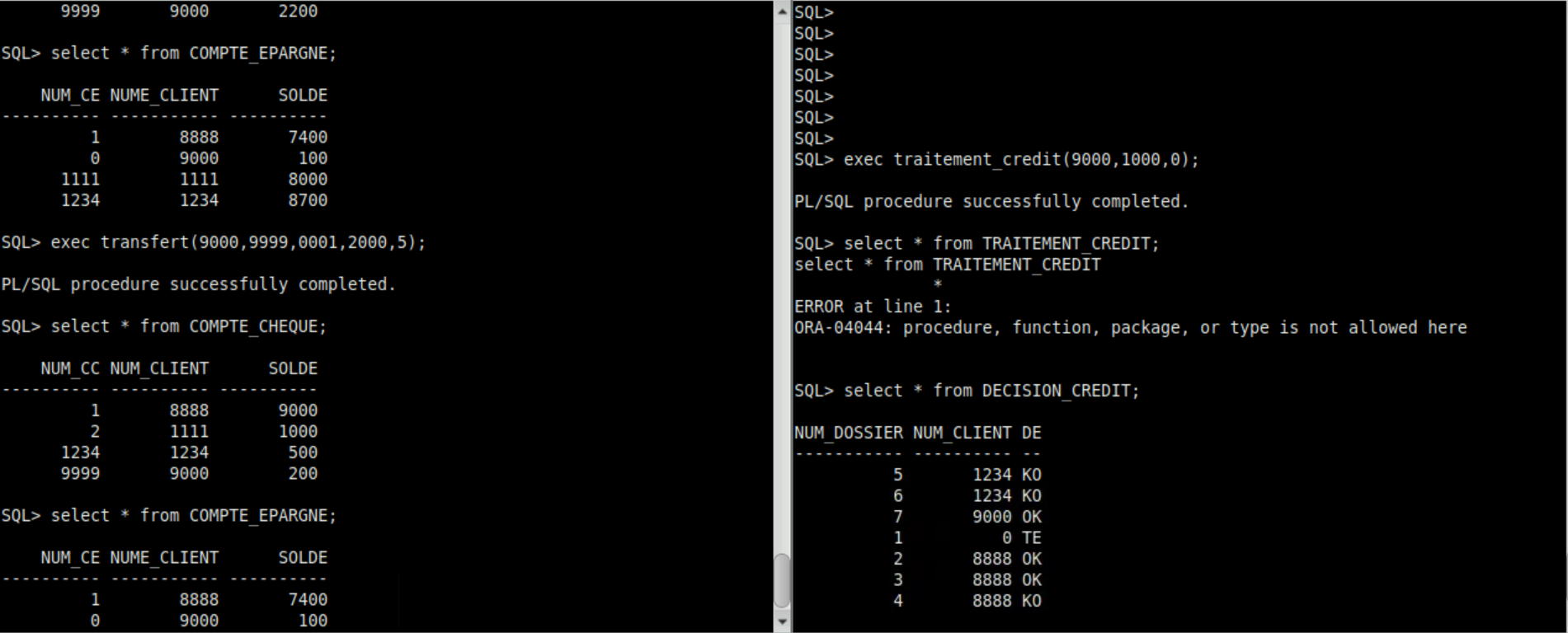
Session 1 Session 2



Un compte tiers transfert de l’argent sur un compte chèque qui, en même temps, envoie de l’argent sur son compte épargne, problème de concurrence.

Transfert d’argent ET demande de crédit :

Session 1 Session 2



Le client numéro 9000 avait un solde tous compte de 2200+100 = 2300 unités.

A ce moment il pouvait emprunter 1000 unités. Il a ensuite fait un virement de 2000 unité au compte chèque numéro 0001, son solde tous compte est donc passé à 200+100 = 300 unités. Il à demandé à emprunté 1000 unités et comme le montre la 3ème ligne de la table DECISION\_CREDIT, ça lui a été accepté alors qu’il possède moins de 3x la somme qu’il souhaite emprunter, problème de concurrence.

Exercice 4 :

Sous Oracle, les instructions entrainent déjà des verrous :

|  |  |
| --- | --- |
| Commande SQL | Mode de Verrou |
| SELECT … FROM table … | Aucun verrou |
| INSERT INTO table … | RX |
| UPDATE TABLE … | RX |
| UPDATE table … | RX |
| DELETE FROM … | RX |

Où : **RX** = Row Exclusive

**SRX** = Share Row Exclusive

**X** = Exclusive (Mode le plus restrictif)

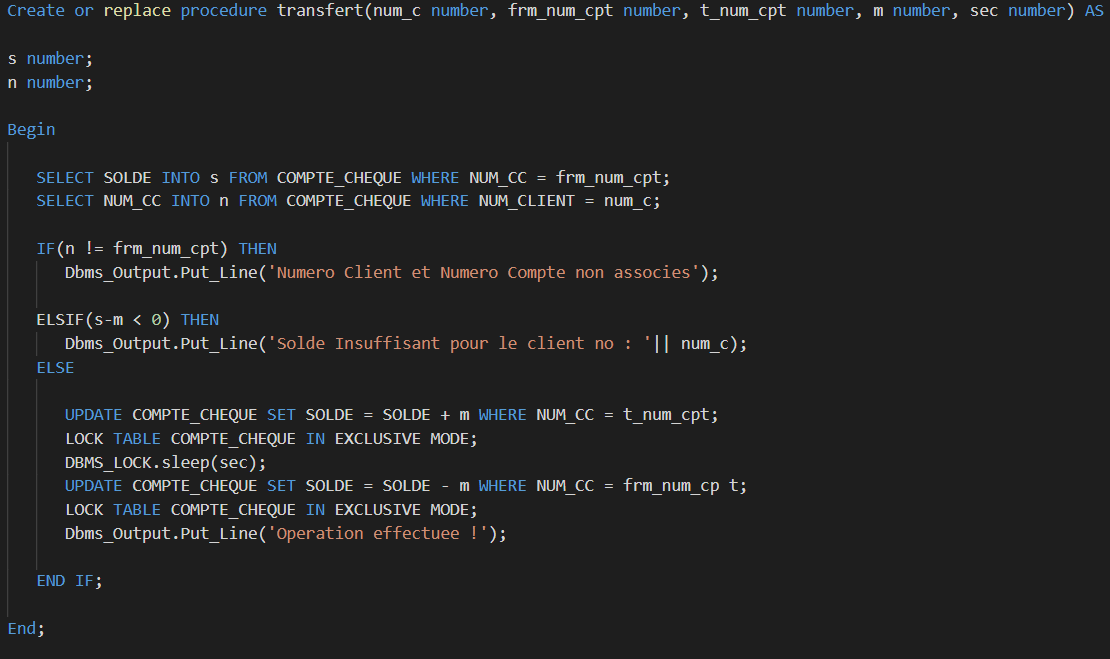
Cependant, j’ai quand même modifié le verrou des changements les plus sensibles (**UPDATE**) en verrou Exclusif (LOCK TABLE nom\_table IN EXCLUSIVE MODE).

Exercice 5 :

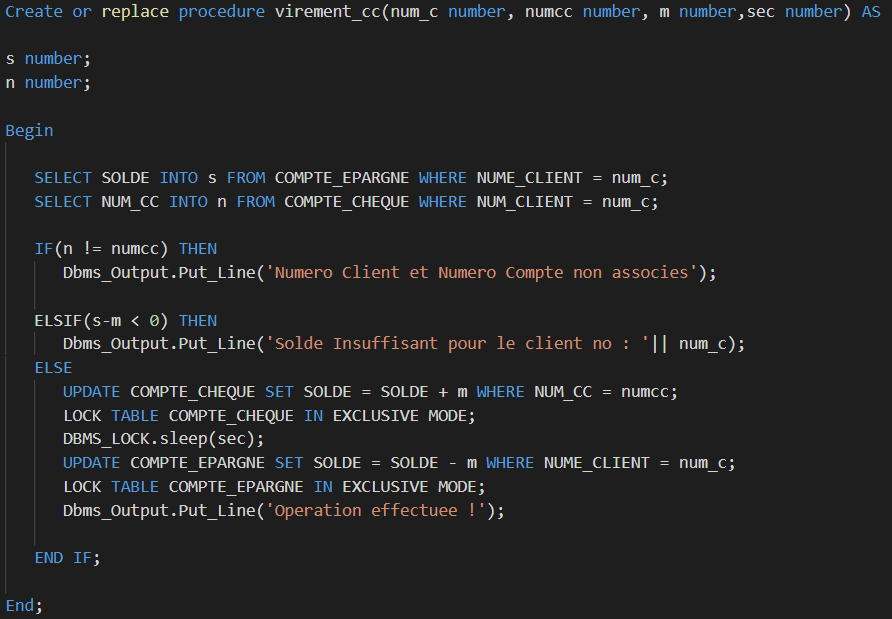
Il suffit d’ajouter un **COMMIT** à la fin des procédures pour éviter ces problèmes de concurrence.

**Code final :**

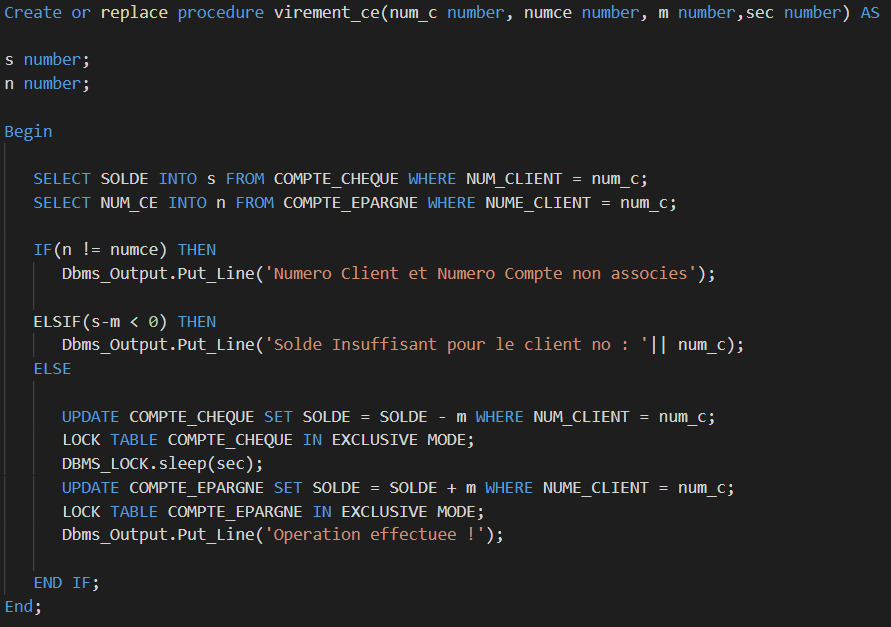
transfert :



virement\_cc :



virement\_ce :



traitement\_credit :

