Nom Prénom

**IBD / TP3 Oracle**

Table des matières

[Table des matières 2](#_Toc56426291)

[1. Introduction 3](#_Toc56426292)

[2. Atomicité 3](#_Toc56426293)

[3. Durabilité 4](#_Toc56426294)

[4. Isolation 5](#_Toc56426295)

[5. Concurrence 6](#_Toc56426296)

[5.1. Lecture/Lecture 6](#_Toc56426297)

[5.2. Ecriture/Ecriture 7](#_Toc56426298)

[5.3. Interblocage 8](#_Toc56426299)

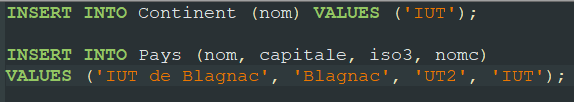
[6. Conclusion 9](#_Toc56426300)

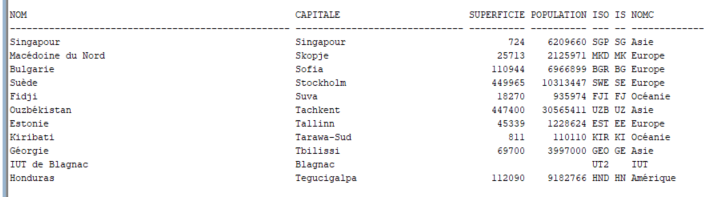
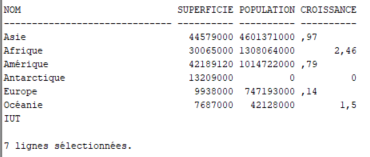
# Introduction

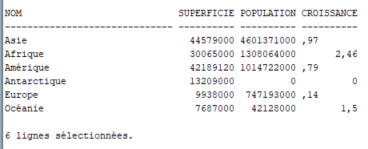
Dans ce TP, nous allons voir les différentes propriétés des transactions lors des manipulations des bases de données. Les 4 propriétés sont : l’atomicité, la cohérence, l’isolation et la durabilité.

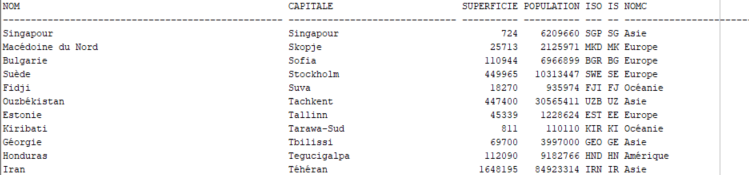
# Atomicité

Pour comprendre cette règle, nous allons commencer par faire une transaction en ajoutant un continent et un pays.

**



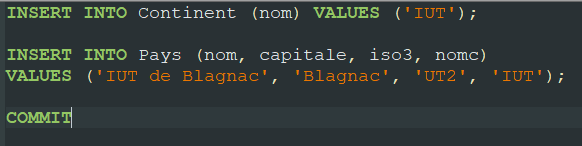
Nous pouvons donc voir la présence des deux éléments ajoutés. Nous allons maintenant les supprimé avec la commande ROLLBACK**

**

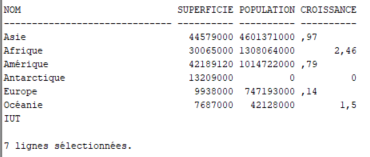
Les deux ajouts ont bien été supprimés. Nous pouvons donc dire que la transaction respecte bien le principe d’atomicité car il faut que l’utilisateur valide les transactions avec la commande COMMIT pour qu’elles prennent effet dans la base de données. Il n’y a pas que l’une des deux qui est validé.

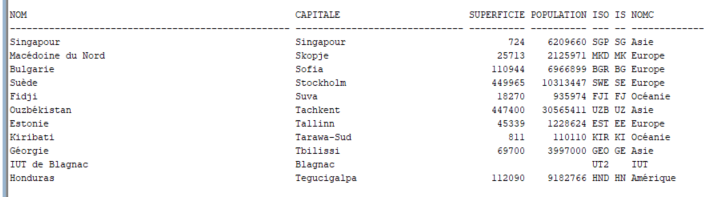
# Durabilité

Nous allons exécuter les mêmes étapes mais cette fois-ci en validant les transactions avec COMMIT



Maintenant, nous nous déconnectons et reconnectons pour vérifier la présence des données.

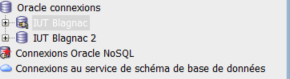




Les deux ajouts sont bien présents après reconnexion. La transaction respecte donc bien le principe de durabilité car les ajouts effectués sont bien présent après fermeture du SGBD.

# Isolation

Nous commençons par créer une nouvelle connexion au serveur de l’IUT de Blagnac dans le logiciel SQL Developper possédant les mêmes paramètres.



Nous allons donc effectués une série d’instruction sur les différentes connexions.

|  |  |
| --- | --- |
| Connexion 1 | Connexion 2 |
| On ajoute les deux pays |  |
| Les deux pays sont présents sur connexion 1 |  |
|  | Mais absent sur connexion 2. En effet, les transactions sur connexion 1 n’étant pas validées, elles n’apparaissent pas sur connexion 2. |
| On alide a présent les modifications |  |
|  | Les deux ajouts sont maintenant présents sur connexion 2 après validation. |
| On supprime ces deux pays |  |
| Les deux pays ont bien été supprimés. |  |
|  | Les deux pays sont toujours présents sur connexion 2. Les suppressions sur connexion 1 n’ont pas été validées. |
|  |  |
|  | Les deux pays ont correctement été supprimés de connexion 2 après validation. |

Nous pouvons donc voir que le principe d’isolation est bien respecté car une session ne peut interagir avec une autre le temps que les transactions n’ont pas été validées.

# Concurrence

## Lecture/Lecture

|  |  |
| --- | --- |
| Connexion 1 | Connexion 2 |
| On ajoute le pays IUT de Rangueil |  |
|  | On ajoute le pays IUT de Rodez |
| TR1  Seul l’IUT de Rangeuil apparait car la transaction de connexion 2 n’a pas été validée |  |
|  | TR2  Seul l’IUT de Rodez apparait car la transaction de connexion 1 n’a pas été validée |
|  |  |
|  | TR2  L’IUT de Rangeuil apparait maintenant après la validation sur connexion 1. |
|  |  |
| L’IUT de Rodez apparait sur connexion 1 après la validation sur connexion 2. |  |

Nous pouvons voir qu’une connexion ne peut pas lire les modifications d’une autre sans qu’elles ne soient validées.

## Ecriture/Ecriture

|  |  |
| --- | --- |
| Connexion 1 | Connexion 2 |
| On supprime les deux pays |  |
|  | On ajoute le pays IUT de Rodez    Le pays ne peut pas être ajouté car pour la connexion 2, il n’a pas encore été supprimé. Le script tourne donc a l’infini    TR1  TR1 |
|  | Le scripte s’exécute avec succès    TR2 |
|  |  |

Nous pouvons donc voir que lorsque deux connexions essayent d’écrire sur une même donnée, la commande de la deuxième connexion ne s’exécutera pas temps que celle de la première n’a pas été validée.

## Interblocage

|  |  |
| --- | --- |
| Connexion 1 | Connexion 2 |
| On ajoute la pays « IUT de Ranguei l » et on valide |  |
|  | On ajoute le pays « IUT de Rodez » et on valide |
| On change le code iso3 de « IUT de Rangueil ». |  |
|  | On change le code iso2 de « IUT de Rodez ». On pose donc un verrou sur ce Pays.    TR2  TR1 |
| Le script tourne à l’ infini. En effet, il y a un verrou sur « IUT de Rodez » par la connexion 2 donc la connexion 1 ne peut pas le modifier. |  |
|  | Le scripte tourne a l’infinie pour les mêmes raisons que la transaction précédente. |
| TR2  Il y a un verrou sur IUT de Rodez |  |
|  | Il n’y a plus de verrou. |

# Conclusion

Dans ce TP, nous avons pu voir les différentes règles des transactions en SQL. En premier l’atomicité avec l’utilisation du commit. En second la durabilité avec la sauvegarde des données après déconnexion. En 3ème l’isolation avec l’utilisation de deux connexion différentes à un même utilisateur et pour finir la cohérence avec els tests d’écriture et de lecture sur les différentes connexions.