ECOLE MAROCAINE DES SCIENCES DE L'INGENIEUR

Membre de HONORIS UNITED UNIVERSITIES

École Marocaine des Sciences de l'Ingénieur

Filière: IIR

Les Bases de Données

La gestion des transactions et le verrouillage

Les Transactions

La Gestion des Transactions dans Oracle

La gestion des transactions est un aspect crucial des bases de données Oracle, permettant de maintenir la cohérence et l'intégrité des données. Dans Oracle, les transactions sont gérées à travers l'utilisation de plusieurs commandes clés, notamment **COMMIT**, **ROLLBACK** et **SAVEPOINT**.

La Gestion des Transactions dans Oracle

Utilisation de COMMIT

La commande **COMMIT** est utilisée pour valider définitivement toutes les modifications effectuées dans la transaction en cours. Une fois que **COMMIT** est exécuté, les modifications sont permanentes et ne peuvent plus être annulées.

Exemple :

```
-- Mise à jour des données
UPDATE EMPLOYEES SET SALARY = SALARY * 1.1 WHERE DEPARTMENT_ID = 30;

-- Validation des modifications
COMMIT;
```

La Gestion des Transactions dans Oracle

Utilisation de ROLLBACK

ROLLBACK est utilisé pour annuler toutes les modifications non validées dans la transaction en cours, toutes les modifications depuis le début de la transaction seront annulées.

Exemple :

```
-- Mise à jour des données
UPDATE EMPLOYEES SET SALARY = SALARY * 1.05 WHERE DEPARTMENT_ID = 30;

-- Validation des modifications
ROLLBACK;
```

La Gestion des Transactions dans Oracle

Utilisation de SAVEPOINT

La commande **SAVEPOINT** permet de définir un point dans une transaction où vous pouvez revenir ultérieurement si nécessaire. Cela est particulièrement utile dans de longues transactions, où vous pourriez vouloir annuler une partie de la transaction sans affecter l'ensemble.

Exemple :

```
-- Début de la transaction
UPDATE EMPLOYEES SET SALARY = SALARY * 1.1 WHERE DEPARTMENT_ID = 30;
SAVEPOINT sp1;              <-- Création d'un SAVEPOINT
UPDATE EMPLOYEES SET SALARY = SALARY * 1.05 WHERE DEPARTMENT_ID = 30;
ROLLBACK TO sp1;            <-- Si nécessaire, vous pouvez revenir au SAVEPOINT

-- La première mise à jour est maintenue, la deuxième annulée
```

La Gestion des Transactions dans Oracle

Exemple de SAVEPOINT :

```
    -- Mise à jour de l'adresse d'un employé
UPDATE EMPLOYEES SET ADDRESS = '123 Bd Sebta' WHERE EMPLOYEE_ID = 114;
SAVEPOINT X1;
    -- Mise à jour du numéro de téléphone de l'employé
UPDATE EMPLOYEES SET PHONE_NUMBER = ‘661-225-555' WHERE EMPLOYEE_ID = 114;
    -- Si la mise à jour du numéro de téléphone doit être annulée
ROLLBACK TO X1;
    -- La mise à jour de l'adresse est conservée
COMMIT;
```

Les Transactions

Exemple Pratique dans le Schéma HR

Démonstration de l'utilisation de COMMIT et ROLLBACK.

Exemple:

```
UPDATE HR.EMPLOYEES SET SALARY = SALARY * 1.1 WHERE DEPARTMENT_ID = 60;

SAVEPOINT sal_increase;

UPDATE HR.EMPLOYEES SET SALARY = SALARY * 1.05 WHERE DEPARTMENT_ID = 90;

ROLLBACK TO sal_increase;          <---- Annule la deuxième mise à jour

COMMIT;                            <------  Valide la première mise à jour
```

Gestion des Verrous dans Oracle

Gestion des Verrous dans Oracle

Les verrous dans Oracle, un système de gestion de base de données, sont un mécanisme crucial pour assurer la cohérence des données et éviter les conflits lors des accès simultanés aux données par plusieurs utilisateurs ou processus.

Oracle utilise deux modes de verrouillage :

- **Exclusif (Exclusive Lock)**: Empêche toute autre transaction de lire ou de modifier les données verrouillées.

- **Partagé (Shared Lock)**: Permet à plusieurs transactions de lire les mêmes données simultanément, mais empêche toute modification de ces données.

Types de Verrous

Oracle utilise principalement deux types de verrous :

1) **Verrous de Table (Table Locks)** : Ils verrouillent une table entière, empêchant d'autres transactions de modifier ou parfois de lire la table.

2) **Verrous de Ligne (Row Locks)** : Ils verrouillent une ligne spécifique dans une table pour empêcher les modifications concurrentes.

Verrous de Table (Table Locks) :

Ces verrous de table s'appliquent explicitement pour contrôler l'accès à une table entière.

Application des verrous de table :

- **Verrouillage en Mode Exclusif (Exclusive Locks)** : Empêche d'autres transactions de lire ou de modifier les données.

```
LOCK TABLE ma_table IN EXCLUSIVE MODE [NOWAIT / WAIT [timeout];
```

- **Verrouillage en Mode Partagé (Shared Locks)** : Permet la lecture mais bloque les modifications concurrentes.

```
LOCK TABLE ma_table IN SHARE MODE [NOWAIT / WAIT [timeout];
```

les verrous de table sont libérés lors du commit ou du rollback.

Verrous de Ligne (Row Locks) :

Ces verrous s'appliquent implicitement lors d'opérations de mise à jour (UPDATE), d'insertion (INSERT) ou de suppression (DELETE).

Application des verrous de table :

1) **Verrouillage en Mode Exclusif (Exclusive Locks)**: Oracle permet également d'appliquer des verrous explicites sur les lignes à l'aide de la clause **FOR UPDATE** dans une requête **SELECT**. Comme montrée ci-dessous :

```
SELECT column_list FROM table_name
WHERE condition FOR UPDATE [NOWAIT | WAIT [timeout]];
```

2) **Verrouillage en Mode Partagé (Shared Locks)**: Oracle ne permet pas d'appliquer ce mode de verrouillage dans les verrous de ligne.

les verrous de table sont libérés lors du commit ou du rollback.

Les Deadlocks

Un **deadlock** est quand deux transactions (ensemble d'opérations de LMD) dans une base de données sont bloquées. Chaque transaction attend que l'autre libère une ressource, comme un enregistrement dans une table, mais aucune ne le fait, et elles restent bloquées indéfiniment.

Comment Oracle s'occupe des Deadlocks

Oracle détecte automatiquement quand il y a un deadlock et choisit une transaction pour l'arrêter et libérer ses ressources. Cela permet aux autres de continuer.

Les Deadlocks

Comment Voir les Deadlocks et les Verrous

Vous pouvez utiliser des commandes spéciales pour voir où il y a des deadlocks et qui attend quoi.

Pour consulter les transactions bloquées:

```
SELECT * FROM V$LOCK WHERE BLOCK=1;
```

Dans Oracle, la colonne BLOCK dans la vue **V$LOCK** indique si un verrou est un "**verrou bloquant**". Si **BLOCK = 1**, cela signifie que la transaction associée à ce verrou est en train de bloquer une autre transaction.

Arréter une transaction bloquée

Si vous savez qu'une transaction cause un problème, vous pouvez l'arrêter avec :

```
ALTER SYSTEM KILL SESSION 'numéro_de_session, numéro_séquentiel';
```

Pour extraire : 'numéro_de_session, numéro_séquentiel

```
SELECT sid, serial#, username, status FROM v$session;
```

Bonnes Pratiques pour Éviter les Deadlocks

- **Transactions Courtes :** Gardez vos transactions aussi courtes et rapides que possible pour réduire le risque de verrous croisés.
- **Verrouillage au Niveau des Lignes :** Utilisez le verrouillage au niveau des lignes plutôt que sur des tables entières pour minimiser l'étendue des verrous
- **Ordre Conséquent des Verrous :** Accédez aux ressources dans la base de données toujours dans le même ordre pour éviter les situations où deux processus attendent l'un sur l'autre.