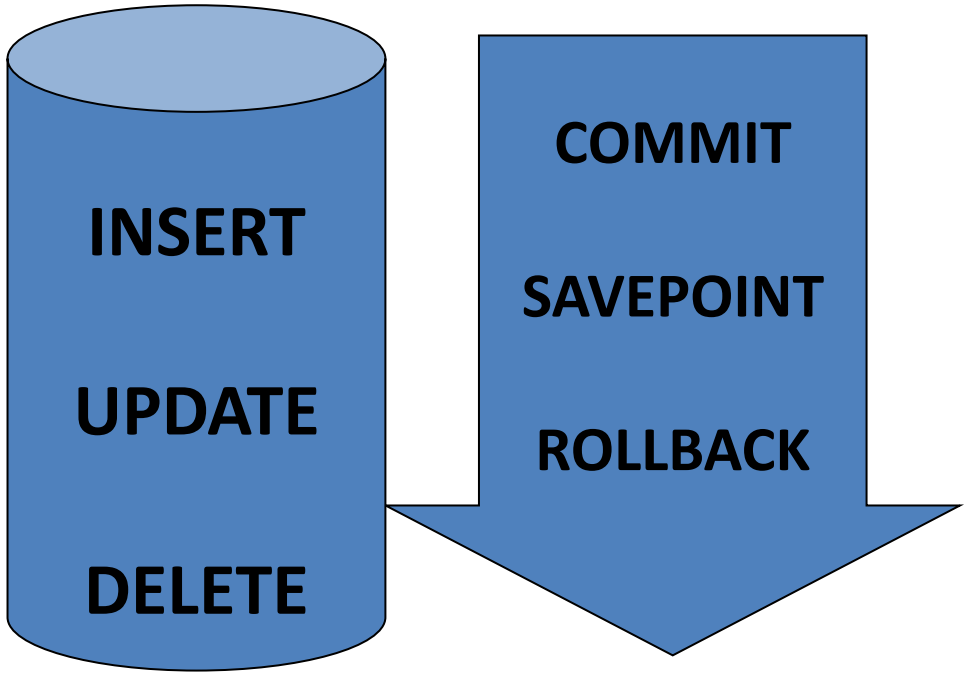
Les Transactions

A tomicité

C ohérence

I solation

D urabilité



Manipulation des données

Pour stocker, modifier, ou supprimer des données dans une base de données Oracle, Vous devez utiliser les commandes du langage des manipulations de données(LMD) qui est un sous langage du SQL.

Ces commandes sont :

- **INSERT** : Permet d'ajouter une ligne dans une table
- **UPDATE** : Permet de modifier des lignes d'une table
- **DELETE** : Permet de supprimer des lignes d'une table

- Pour exécuter des commandes LMD, l'utilisateur doit disposer des privilèges INSERT, UPDATE et DELETE.
- Ces privilèges sont accordés automatiquement au propriétaire de la table concernée par les modifications, mais ils doivent être accordés explicitement aux autres utilisateurs.

- Une transaction est un ensemble d'instructions LMD de SQL qui ont pour objectif de faire passer la base de données, en une seule étape, d'un état cohérent à un autre état cohérent.
- Une transaction qui réussit, modifie la base de données dans un nouvel état cohérent.
- Si une transaction échoue (volontairement ou involontairement), les modifications déjà effectuées dans la base sont annulées, de sorte qu'elle retrouve l'état cohérent antérieur au début de la transaction.
  - C'est Oracle qui se charge entièrement de toute cette gestion.

- Une base de données ne prenant en charge qu'un seul utilisateur n'est pas très utile.
- Le contrôle de multiples utilisateurs mettant à jour les mêmes données et en même temps est crucial.
  - Il est lié à l'uniformité et à la simultanéité des données
- L'uniformité des données signifie que les résultats visualisés par une personne sont cohérents à l'intérieur d'une ou plusieurs transactions courantes.
- La simultanéité des données signifie que de nombreuses personnes peuvent accéder aux mêmes données en même temps.

- Les transactions, composées d'une suite d'opérations, gagnent à être aussi petites que possible.
- Afin qu'une série d'instructions SQL soit considérée comme une transaction, elle doit présenter les propriétés suivantes:
- **Atomicité** : Une transaction doit être une unité atomique de travail; elle ne peut réussir que si toutes ses opérations réussissent.
- **Cohérence** : Quand une transaction est terminée, elle doit laisser les données dans un état cohérent incluant toutes les règles d'intégrité de données.

- **Isolation** : les transactions doivent être isolées des changements effectués par d'autres transactions.
  Le niveau d'isolation est configurable par l'application.
- **Durabilité** : une transaction doit être récupérable aussitôt quelle est terminée.
  Même si un échec du système se produit après la fin de la transaction, les effets de la transaction sont permanents dans le système.

- Les commandes SQL utilisées pour contrôler les transactions dans une base de données sont appelées des commandes de contrôle de transaction.
- Il existe trois commandes de contrôle de transaction:
- **COMMIT** : Rend permanentes toutes les modifications effectuées provisoirement dans une transaction.
- **ROLLBACK** : Permet d'annuler des modifications en attente de validation.
- **SAVEPOINT** : Définit une étiquette à laquelle vous pouvez limiter une opération de ROLLBACK permettant ainsi d'annuler des modifications jusqu'à un point particulier défini par cette étiquette.

Insertion de nouvelles de données dans une table

INSTRUCTION INSERT

Pour ajouter de nouvelles lignes de données à une table d'une base de données , vous pouvez utiliser la commande **INSERT** dont la syntaxe est :

INSERT INTO ***table [(colonne [,colonne] …)]*** **VALUES** ***(valeur [, valeur] …]);***

table : spécifie le nom de la table à laquelle vous souhaitez ajouter une ligne.

[(colonne [,colonne]…) : spécifie la liste des noms des colonnes de la table auxquelles les valeurs que vous insérez sont destinées. En absence de la liste, le serveur considère que vous allez fournir dans la clause VALUES une valeur pour chaque colonne, en respectant l'ordre naturel.

(valeur [, valeur] …]): spécifie la liste des valeurs à insérer dans les colonnes correspondantes de la table.

Quelques règles à respecter lors de l'ajout de nouvelles lignes

Vous devez fournir les valeurs selon l'ordre par défaut des colonnes de la table, sauf si un autre ordre est précisé dans le paramètre définissant la liste des colonnes.

Les valeurs chaînes de caractères et dates doivent être placées entre apostrophes

Si vous ne précisez pas la valeur des dernières colonnes, ORACLE les traite comme si elles contenaient la valeur NULL.

Il est possible d'insérer une valeur NULL explicite dans une table en spécifiant le mot-clé NULL dans la liste de valeurs.

Pour les colonnes contenant des dates et des caractères, vous pouvez utilisez la notation de chaîne vide (‘’) dans la liste VALUES pour définir une valeur NULL.

Avant d'insérer une valeur NULL dans une colonne, vérifiez que la colonne de destination accepte ce type de valeur. Pour ce faire, utilisez la commande SQL*PLUS DESCRIBE

Exemples d'insertion de données

```
INSERT  INTO  Departement (numDept, nom, ville)
                    VALUES  (50, 'Informatique', NULL)
```

L'exemple ci-dessus ajoute une ligne à la table Departement. La colonne numDept reçoit la valeur 50, la colonne nom reçoit la valeur 'Informatique', et la colonne ville reçoit la valeur NULL.

Lorsque vous ajoutez des lignes à une table, vous n'avez pas toujours besoin de toutes les colonnes de la table. Si vous voulez spécifier des valeurs pour certaines colonnes seulement, vous pouvez utiliser la liste de colonnes dans la clause INTO

La liste de colonnes peut également servir à définir l'ordre dans lequel les valeurs sont fournies dans la clause VALUES. Par exemple, l'instruction ci-dessous ajoute une nouvelle ligne dans laquelle la colonne nom reçoit la valeur 'Marketing', et la colonne numDept reçoit la valeur 60.

```
INSERT  INTO  Departement (nom, numDept)  VALUES  ('Marketing', 60)
```

Si vous ne spécifiez pas de liste de colonnes, vous devez fournir explicitement la valeur de chacune des colonnes de la table, dans l'ordre où elles ont été créées.

L'instruction ci-dessous ne spécifie pas la liste de colonnes puisqu'elle fournit une valeur à chaque colonne de la table Departement.

```
INSERT INTO Departement VALUES (70, 'Recherche', 'Fes');
```

Insertion des données de type INTERVAL YEAR TO MONTH

```
INSERT INTO CourseAPied (numCourse, pays, t_s_d_edition)
VALUES (10, 'Monde', INTERVAL '123-2' YEAR(3) TO MONTH );

INSERT INTO CourseAPied (numCourse, pays, t_s_d_edition)
VALUES (10, 'Monde', INTERVAL '123' YEAR(3) );

INSERT INTO CourseAPied (numCourse, pays, t_s_d_edition)
VALUES (10, 'Afrique', INTERVAL '4' YEAR );

INSERT INTO CourseAPied (numCourse, pays, t_s_d_edition)
VALUES (10, 'Afrique', INTERVAL '300' MONTH(3));

INSERT INTO CourseAPied (numCourse, pays, t_s_d_edition)
VALUES (10, 'Europe', INTERVAL '50' MONTH);
```

Insertion des données de type INTERVAL DAY TO SECOND

```
INSERT INTO CourseAPied (numCourse, recordDistance,tempsVainqueur)
VALUES (20, INTERVAL '4 5:12' DAY TO MINUTE, INTERVAL '4 5:12:10.222' DAY TO
SECOND(3) )
```

```
INSERT INTO CourseAPied (numCourse, recordDistance,tempsVainqueur)
VALUES (20, INTERVAL '400' DAY(3), INTERVAL '400 5' DAY(3) TO HOUR)
```

```
INSERT INTO CourseAPied (numCourse, recordDistance,tempsVainqueur)
VALUES (20, INTERVAL '11:20' HOUR TO MINUTE, INTERVAL '11:12:10.2222222' HOUR
TO SECOND(7))
```

```
INSERT INTO CourseAPied (numCourse, recordDistance,tempsVainqueur)
VALUES (20, INTERVAL '10' MINUTE, INTERVAL '10:22' MINUTE TO SECOND)
```

```
INSERT INTO CourseAPied (numCourse, recordDistance,tempsVainqueur)
VALUES (20, INTERVAL '30' SECONDE, INTERVAL '30.12345' SECOND(2,4))
```

Insertion de valeurs spéciales

Il est souvent nécessaire d'enregistrer la date ou l'heure courante comme valeur spéciale dans une table.

De la même façon, il peut s'avérer utile de stocker le nom de l'utilisateur Oracle en cours dans une colonne.

Oracle fournit des fonctions qui renvoient des valeurs spéciales concernant l'environnement courant telles que SYSDATE et USER.

La fonction USER renvoie le nom de l'utilisateur en cours, et peut être utilisée dans l'instruction INSERT pour insérer le nom de l'utilisateur courant dans une colonne de type caractère.

La fonction SYSDATE renvoie la date et l'heure en cours au format de date Oracle.

L'instruction ci-dessous insère une ligne dans la table Employe.
La colonne numEmp reçoit la valeur 10, nom reçoit le nom de l'utilisateur courant, et dateEmb reçoit la date courante.

```
INSERT INTO Employe (numEmp, nom, dateEmb)
VALUES(10, USER, SYSDATE)
```

La fonction de conversion SQL TO_DATE est également commode pour fournir des valeurs lors de l'insertion de lignes. Par défaut, le serveur attend des valeurs de date au format Oracle standard, JJ-MM-AAAA.

Si vous devez utiliser un autre format de la date, vous devez utiliser la fonction TO_DATE pour décrire les éléments du format que vous fournissez.
L'instruction ci-dessous utilise la fonction TO_DATE pour spécifier un format de date différent de celui par défaut.

```
INSERT INTO Employe (numEmp, nom, dateEmb) VALUES
(20, USER, TO_DATE('01-jan-1983 11:30:20', 'DD-MON-YYYY  HH24:MI:SS')
```

Ajout à une table des lignes provenant d'une autre table

Il est possible d'ajouter à une table des lignes de données extraites d'une autre table, en utilisant la commande INSERT avec la syntaxe ci-dessous:

INSERT INTO *table [(colonne [,colonne] …)]* <sous requete>;

Cette syntaxe n'utilise pas la clause VALUES, mais utilise à la place une sous-requête.

Une sous-requête est une instruction SELECT incorporée qui permet de sélectionner des lignes dans une ou plusieurs tables et, en l'occurrence, les copie dans la table où vous insérez des lignes

L’instruction ci-dessous copie dans la table CourseArchive des lignes de la table CourseAPied pour lesquelles la date de départ est inférieure à ’31-DEC-2010’

```
INSERT  INTO  CourseArchive (numCourse, pays, ville, distance, dateDepart)
SELECT numCourse, pays, ville, distance, dateDepart
FROM CourseAPied
WHERE dateDepart < ’31-DEC-2010’
```

Mise à jour des données d'une table

INSTRUCTION UPDATE

Pour mettre à jour les lignes d'une table, il faut utiliser la commande UPDATE dont la syntaxe est :

```
UPDATE table
SET colonne = valeur [, colonne = valeur …]
[WHERE condition];
```

table : spécifie le nom de la table dont les lignes doivent être mises à jour.

colonne = valeur [, colonne = valeur …] : spécifie la ou les colonnes à mettre à jour. Le terme value spécifie la nouvelle valeur à attribuer à la colonne correspondante, et peut être une constante, une expression, ou une sous-requête .

Condition : Identifie les lignes à mettre à jour. Elle limite la portée de la commande aux lignes qui satisfont la condition.

L’instruction ci-dessous modifie la colonne numDept de la table Employe pour l’employé dont le numéro est 5. L’employé est transféré au département 60.

```
UPDATE Employe
SET numDept = 60
WHERE numEmploye = 5;
```

L’instruction ci-dessous modifie deux colonnes de la table Employe pour l’employé de numéro 10. La colonne numDept prend la valeur 50, et le salaire la valeur 15000.

```
UPDATE Employe
SET numDept=50, salaire=15000
WHERE numEmploye = 10;
```

Il est possible de mettre à jour toutes les lignes d'une table en utilisant la commande UPDATE sans la clause WHERE.

Bien que cela puisse parfois s'avérer nécessaires, veillez à ne pas omettre la clause WHERE par inadvertance.

L'instruction ci-dessous augmente de 5% le salaire de tous les employés de la table Employe.

```
UPDATE Employe
SET salaire =  salaire * 5/100
```

Suppression des données d'une table

INSTRUCTION DELETE

Pour supprimer des lignes de données d'une table, vous pouvez utiliser la commande DELETE dont la syntaxe est:

DELETE [FROM] *table*
[WHERE *condition*];

table : spécifie le nom de la table dans laquelle vous souhaitez supprimer des lignes.

condition : spécifie la condition que doivent vérifier les lignes à supprimer. Toute condition SQL est autorisée, y compris une sous requête.

- La clause WHERE est facultative. Si elle est omise, toutes les lignes de la table sont supprimées

- Le mot-clé FROM est facultatif, et sert uniquement à améliorer la lisibilité

Attention aux contraintes d'intégrité. Si vous essayez de supprimer une ligne qui contient une clé primaire utilisée comme clé étrangère dans une autre table, le serveur Oracle signale une erreur de contrainte d'intégrité.

L'instruction ci-dessous supprime des lignes de la table employé qui correspondent aux employés embauchés avant le 1er janvier 1990

```
DELETE  FROM  Employe
WHERE dateEmb < TO_DATE ('01-01-1990', 'DD-MM-YYYY')
```

L'instruction ci-dessous supprime toutes les lignes de la table Employe. Il ne reste donc que la structure de la table.

```
DELETE  FROM  Employe
```

L’instruction ci-dessous supprime toutes les lignes de la table Employe qui correspondent aux employés qui travaillent dans le(s) même(s) département(s) que Moha.

```
DELETE  FROM  Employe
WHERE numDept  IN (SELECT numDept
                   FROM Employe
                   WHERE nom = ‘Moha’);
```

LES TRANSACTIONS

La cohérence des données dans une base de données est assurée par l'utilisation de transactions. Ces dernières fournissent une certaine souplesse et le contrôle nécessaire pour modifier les données. Elles garantissent par ailleurs la cohérence des données en cas d'échec d'un processus.

Une transaction est une séquence d'instructions LMD que Oracle traite comme une seule unité. Une transaction permet donc de valider (COMMIT) ou d'annuler (ROLLBACK) un ensemble d'instructions LMD en tant qu'entité unique.

LES TRANSACTIONS

Une transaction commence à la première instruction SQL exécutable rencontrée.

Lorsqu'une transaction est terminée, l'instruction SQL exécutable suivante démarre automatiquement la transaction suivante.

Plusieurs actions peuvent marquer la fin d'une transaction :

♣ l'utilisation de la commande COMMIT ou ROLLBACK.

♣ L'exécution d'une commande LDD (Langage de Définition de Données)

♣ L'exécution d'une commande LCD (Langage de Contrôle de Données).

♣ La fin d'une session SQLDeveloper.

♣ Panne du système. Dans ce cas, toutes les transactions en attente sont annulées.

Commandes de contrôles des transactions

Il est possible de valider ou d'annuler les modifications en attente au sein d'une transaction à l'aide de commandes SQL appelées commandes de contrôle des transactions.

Ces commandes sont :

1. COMMIT,
2. SAVEPOINT,
3. ROLLBACK.

Commande COMMIT

Cette commande termine la transaction en cours en rendant permanentes les modifications de données effectuées provisoirement. Sa syntaxe est :

```
COMMIT  [WORK] [ { COMMENT 'Text'  | FORCE 'Text' [, integer] } ] ;
```

Le mot-clé WORK est facultatif, et est supporté uniquement pour la compatibilité avec SQL satandard.

La clause COMMENT permet d'associer un commentaire à une transaction. 'Text' est une constante de 255 caractères au plus stocké par Oracle dans la vue DBA_2PC_PENDING du dictionnaire de données.

La clause FORCE est utilisée dans les bases de données distribuées.

Exemples d'utilisation de COMMIT:

Validation d'une instruction d'insertion:

```
INSERT INTO Departement VALUES (70, 'Approvisionnement', 'Casablanca');
COMMIT;
```

Commentaire sur une validation d'une transaction:

```
UPDATE EMPLOYE SET salaire = salaire + salaire * 5/100;
COMMIT  COMMENT 'Augmentation des salaires des employés';
```

Commande SAVEPOINT

Cette commande place une étiquette au sein d'une transaction pour identifier un point auquel on peut effectuer un rollback. Il est possible de placer plusieurs étiquettes dans une même transaction.

La syntaxe de SAVEPOINT est :

```
SAVEPOINT etiquette;
```

Exemple d'utilisation de la commande SAVEPOINT:

```
UPDATE  Employe  SET Salaire = 7000  WHERE nom = 'Moha'
SAVEPOINT  Moha_Sal;
UPDATE  Employe  SET Salaire = 12000  WHERE nom = 'Manolo'
SAVEPOINT  Manolo_Sal;
SELECT SUM (Salaire) FROM Employe;
ROLLBACK TO SAVEPOINT Moha_Sal;
UPDATE  Employe  SET Salaire = 11000  WHERE nom = 'Manolo';
COMMIT;
```

Commande ROLLBACK

Cette commande termine la transaction en cours en annulant toutes les modifications de données effectuées provisoirement et également tous les SAVEPOINTs. La syntaxe est:

```
ROLLBACK [WORK] [ { TO [SAVEPOINT] etiquette | FORCE 'Text' } ] ;
```

Le mot-clé WORK est optionnel et existe uniquement pour la compatibilité avec

SQL Standard.

La clause FORCE est utilisée dans les bases de données distribuées.

La clause TO SAVEPOINT spécifie l'étiquette qui identifie le point vers lequel on veut effectuer un rollback.

Si la clause SAVEPOINT est omise un rollback de toute la transaction est effectué.

UN ROLLBACK SANS SAVEPOINT effectue les opérations suivantes:

♣ Termine la transaction
♣ Annule toutes les modifications effectuées par la transaction
♣ Efface tous les savepoints
♣ Annule tous les verrous posés par la transaction.

UN ROLLBACK AVEC SAVEPOINT effectue les opérations suivantes:

♣ Annule uniquement les modifications entre le savepoint réfèrencé et le ROLLBACK qui y réfère
♣ Efface tous les savepoints entre le savepoint référencé et le ROLLBACK qui y réfère. Le savepoint référencé est conservé, et on peut y référer par un autre ROLLBACK
♣ Annule les verrous posés sur les données modifiées entre le savepoint et le ROLLBACK qui y réfère. Par conséquent:
Les autres transactions qui avaient demandé accès aux données verrouillées après le savepoint doivent attendre jusqu'à ce que la transaction soit validée (committed) ou annulée (rolled back).
Les autres transactions qui n'avaient pas demandé accès aux données verrouillées peuvent accéder aux données immédiatement sans attendre la fin de la transaction.

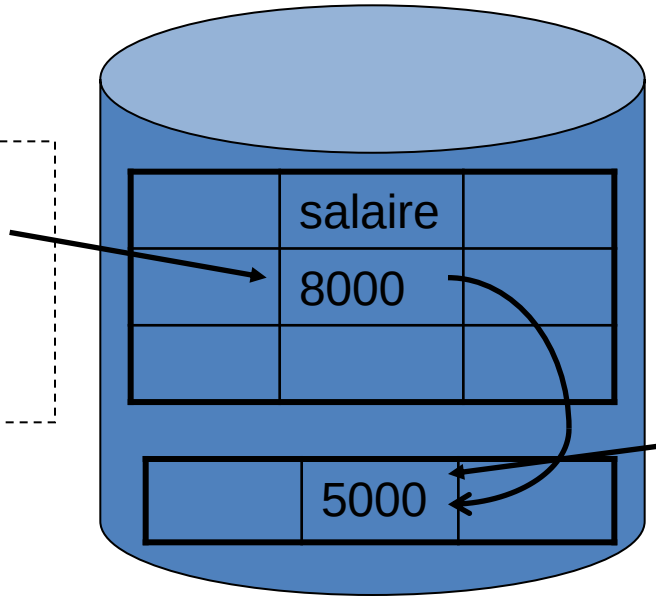
Dans une base de données, la cohérence des données est assurée par l'utilisation de transactions. Une transaction est considérée incomplète tant qu'elle n'est pas validée.

Vous pouvez valider une transaction en émettant la commande COMMIT.
Avant de valider une transaction, vous devez connaître l'état des données avant et après l'exécution de la commande COMMIT.

Lorsque des opérations LMD sont effectuées mais pas encore validées, Oracle conserve provisoirement l'état des données avant modification pour le rétablir automatiquement en cas de rollback.

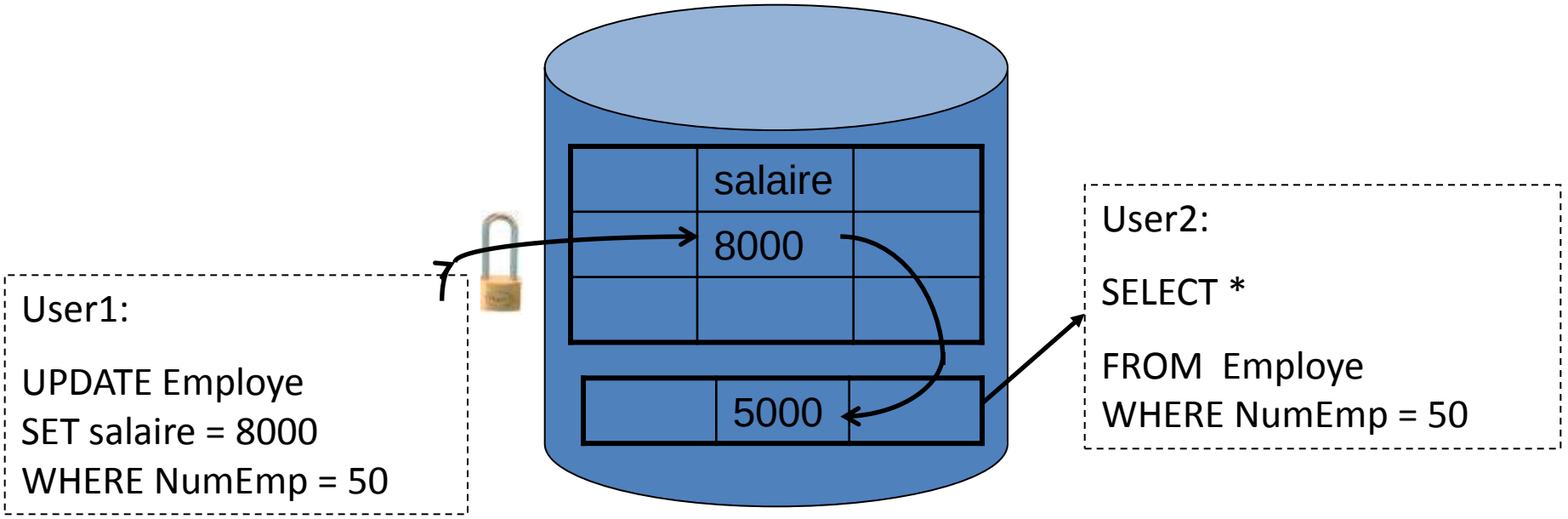
Opération LMD :

UPDATE Employe
SET salaire = 8000
WHERE NumEmp = 50



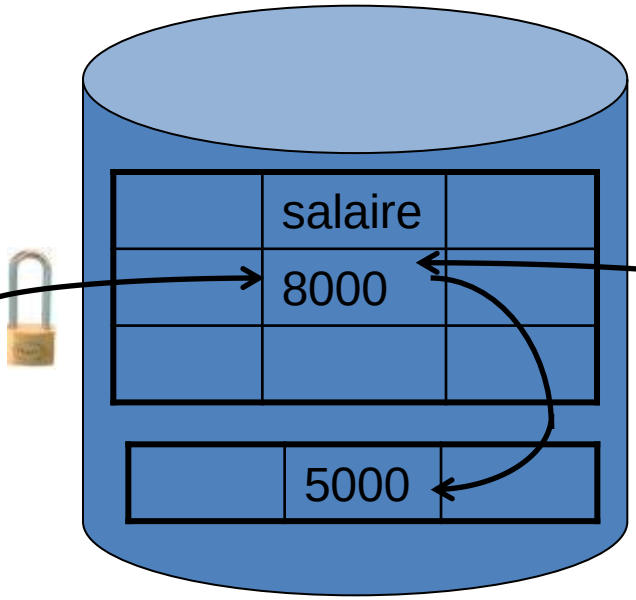
Anciennes valeurs conservées pendant la transaction

Vous pouvez afficher les résultats d'une opération effectuée sur une table avant de valider une transaction.
Pour afficher le résultat d'une instruction LMD, interrogez la table à l'aide de l'instruction SELECT.
Par exemple, dans le graphique représenté à l'écran, deux utilisateurs distincts effectuent des opérations SELECT et UPDATE.



Tant que vous êtes dans une transaction non permanente, les autres utilisateurs ne peuvent pas afficher les résultats des opérations LMD que vous effectuez. Oracle impose la cohérence en lecture pour garantir que chaque utilisateur telles qu'elles existaient lors de la dernière validation. USER2 obtiendra l'ancienne valeur du salaire c'est à 5000.

Lorsque vous effectuez une opération LMD, Oracle verrouille les lignes concernées. Les autres utilisateurs ne peuvent donc pas modifier les données affectées par votre commande LMD.



Non autorisée
Données verrouillées

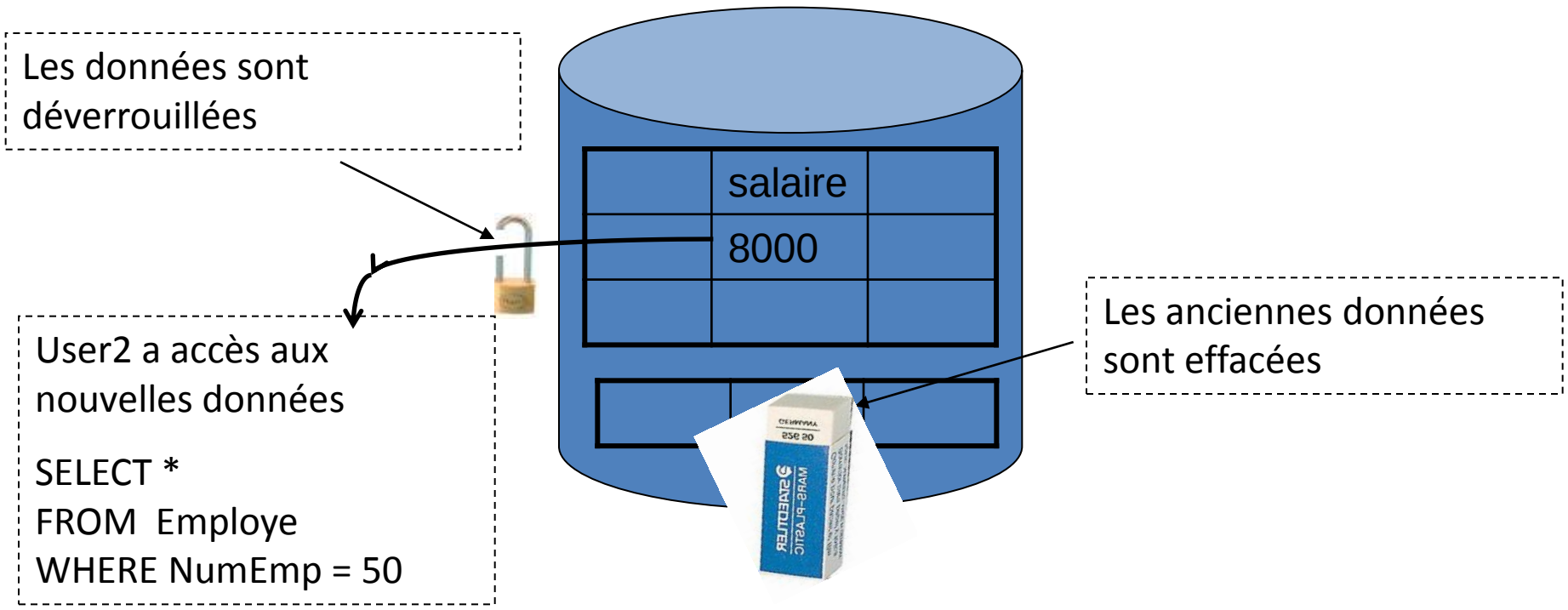
User1:

UPDATE Employe
SET salaire = 8000
WHERE NumEmp = 50

User2:

UPDATE Employe
SET salaire = 15000
WHERE NumEmp = 50

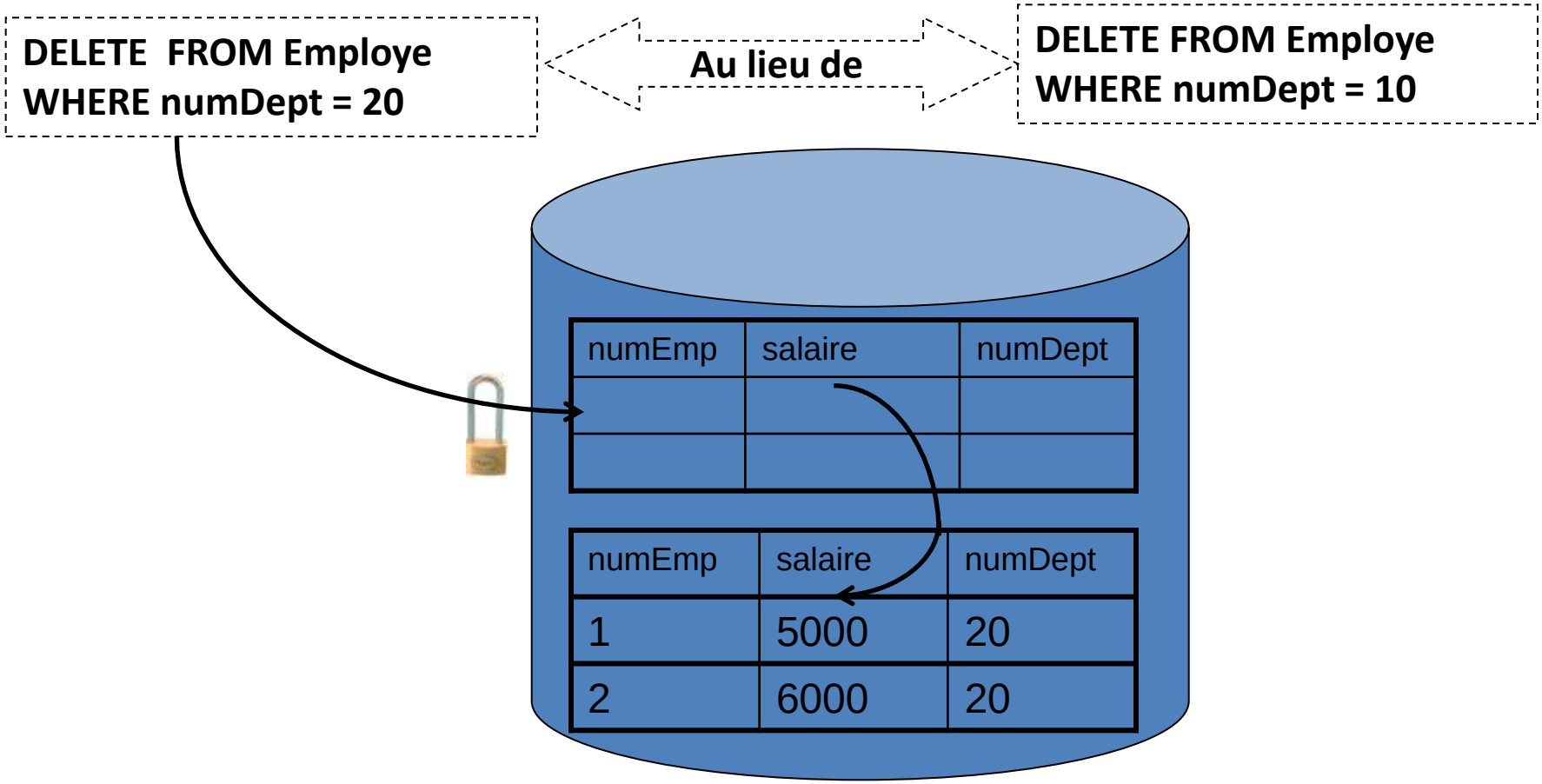
Une fois qu'une transaction est validée, l'état antérieur des données modifiées est supprimé et les modifications effectuées deviennent permanentes dans la base de données.



Lorsqu'une transaction est validée, tous les utilisateurs peuvent afficher ses résultats.
Lorsqu'une transaction est en cours de traitement, Oracle verrouille les lignes concernées.
Ces verrous sont annulés lorsque la transaction est terminée.

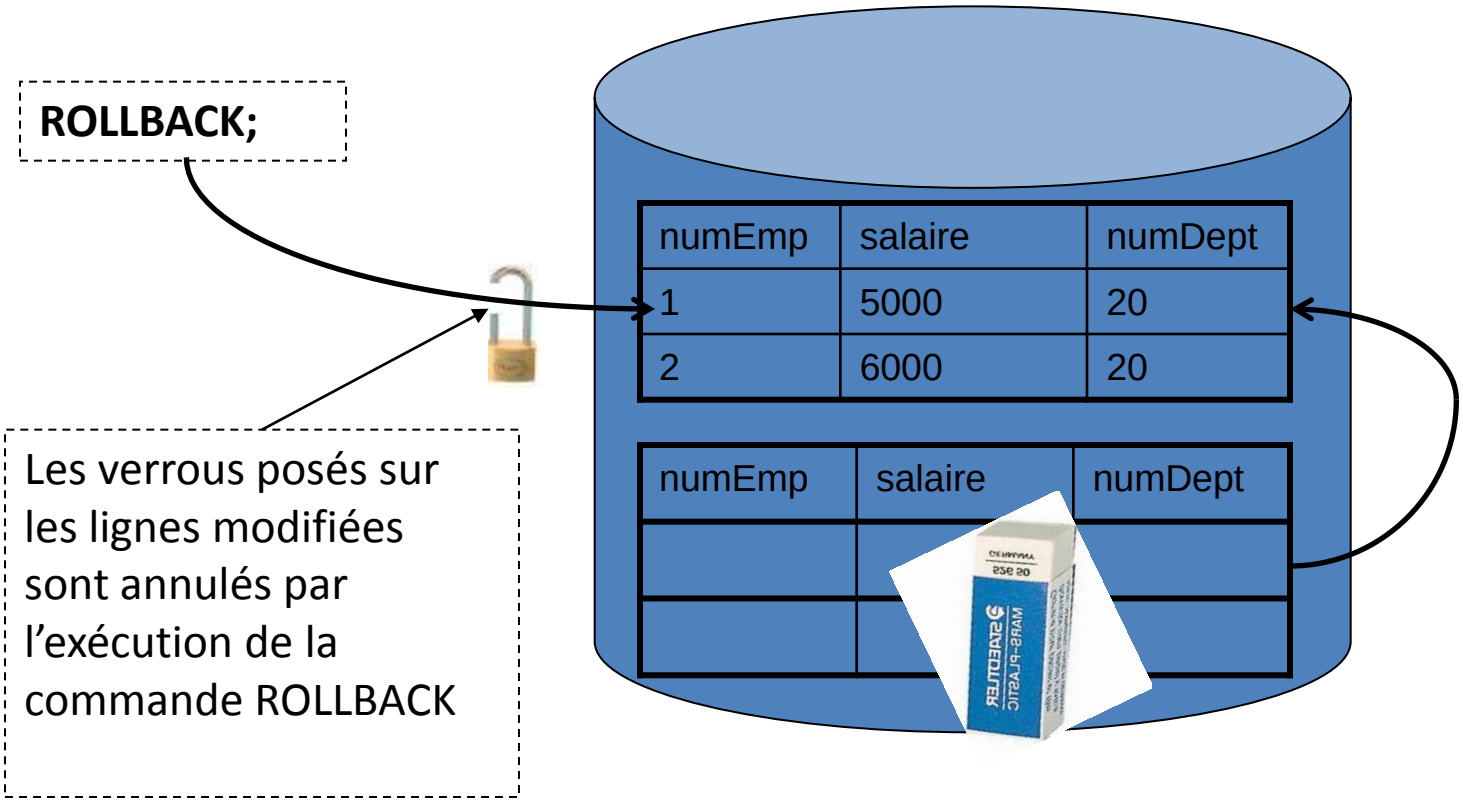
La validation d'une transaction efface tous les savepoints.

Il peut vous arriver de supprimer des lignes par erreur. Supposez par exemple que vous ayez supprimé les employés du département 20 au lieu de ceux du département 10.



Vous pouvez annuler toutes les modifications (ici les suppressions) effectuées provisoirement par la transaction en utilisant la commande ROLLBACK

Lorsqu'une commande ROLLBACK est exécutée, toutes les modifications de données non encore validées sont annulées. Pour ce faire, Oracle rétablit l'ancien état des données qui est conservé jusqu'à la fin de la transaction



INSERT…
INSERT….
INSERT..
SAVEPOINT Insert_fait
UPDATE…
UPDATE….
UPDATE..
SAVEPOINT Update_fait
DELETE…
DELETE….
…….

Il est possible de subdiviser une transaction en petites sections exécutables en créant des savepoints. Ces étiquettes jouent le rôle de bornes limitant l'effet d'une opération de rollback.

Une étiquette est créée par la commande SAVEPOINT

Pour annuler les commandes non validées à partir d'une étiquette, utilisez la commande:

ROLLBACK TO SAVEPOINT etiquette;

Par exemple pour annuler uniquement les commandes DELETE, Exécutez:

ROLLBACK TO SAVEPOINT Update_fait;

Pour annuler à la fois les commandes DELETE et UPDATE, exécutez :

ROLLBACK TO SAVEPOINT Insert_fait;

Pour annuler toutes les commande DELETE, UPDATE et INSERT, exécutez :

ROLLBACK;