Manipuler des données

Langage de manipulation de données

- Une instruction LMD est exécutée dans les cas suivants :
  - Vous ajoutez de nouvelles lignes à une table.
  - Vous modifiez des lignes existantes d'une table.
  - Vous supprimez des lignes existantes d'une table.
- Une *transaction* est composée d'un ensemble d'instructions LMD qui forment une unité de travail logique.

Ajouter une nouvelle ligne à une table

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

	70 Public Relations	100	1700

Nouvelle ligne

Insérez une nouvelle ligne dans la table DEPARTMENTS.

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	70	Public Relations	100	1700
2	10	Administration	200	1700
3	20	Marketing	201	1800
4	50	Shipping	124	1500
5	60	IT	103	1400
6	80	Sales	149	2500
7	90	Executive	100	1700
8	110	Accounting	205	1700
9	190	Contracting	(null)	1700

Syntaxe de l'instruction `INSERT`

- Ajoutez de nouvelles lignes à une table à l'aide de l'instruction `INSERT` :

```
INSERT INTO   table [(column [, column...])]
VALUES     (value [, value...]);
```

- Avec cette syntaxe, une seule ligne est insérée à la fois.

Insérer de nouvelles lignes

- Insérez une nouvelle ligne contenant des valeurs pour chaque colonne.
- Enumérez les valeurs dans l'ordre par défaut des colonnes de la table.
- Enumérez éventuellement les colonnes indiquées dans la clause `INSERT`.

```
INSERT INTO departments(department_id,
       department_name, manager_id, location_id)
VALUES (70, 'Public Relations', 100, 1700);
1 rows inserted
```

- Placez les valeurs de type caractère et date entre apostrophes.

Insérer des lignes comprenant des valeurs NULL

- Méthode implicite : Omettre la colonne dans la liste.

```
INSERT INTO  departments (department_id,
                          department_name)
VALUES     (30, 'Purchasing');
1 rows inserted
```

- Méthode explicite : Indiquer le mot-clé `NULL` dans la clause `VALUES`.

```
INSERT INTO  departments
VALUES     (100, 'Finance', NULL, NULL);
1 rows inserted
```

Insérer des valeurs spéciales

La fonction `SYSDATE` enregistre la date du jour et l'heure actuelle.

```
INSERT INTO employees (employee_id,
                       first_name, last_name,
                       email, phone_number,
                       hire_date, job_id, salary,
                       commission_pct, manager_id,
                       department_id)
VALUES          (113,
                       'Louis', 'Popp',
                       'LPOPP', '515.124.4567',
                       SYSDATE, 'AC_ACCOUNT', 6900,
                       NULL, 205, 110);
```

```
1 rows inserted
```

Insérer des valeurs de date et d'heure spécifiques

- Ajoutez un nouvel employé.

```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
             'SA_REP', 11000, 0.2, 100, 60);
```

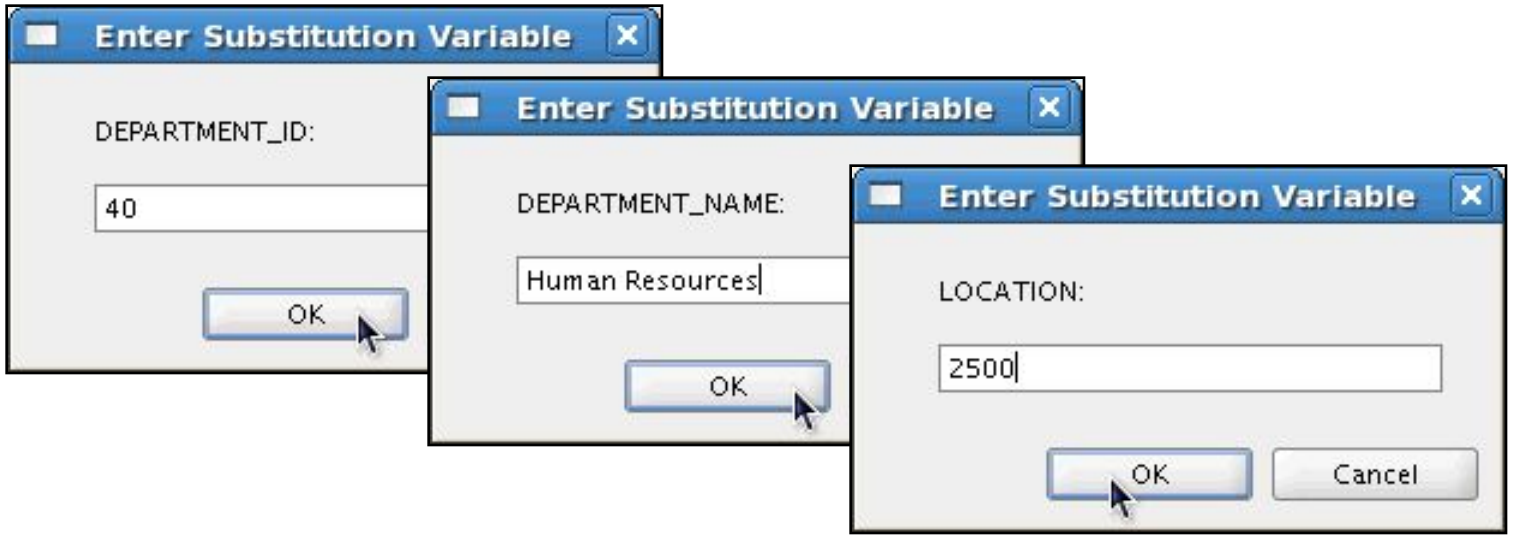
```
1 rows inserted
```

- Vérifiez votre ajout.

Créer un script

- Utilisez une variable de substitution avec esperluette (&) dans une instruction SQL pour afficher une invite de saisie de valeur.
- Une telle variable est un paramètre substituable pour une valeur.

```
INSERT INTO departments
            (department_id, department_name, location_id)
VALUES      (&department_id, '&department_name',&location);
```



Copier des lignes depuis une autre table

- Ecrivez l'instruction `INSERT` avec une sous-interrogation :

```
INSERT INTO sales_reps(id, name, salary, commission_pct)
  SELECT employee_id, last_name, salary, commission_pct
  FROM   employees
  WHERE  job_id LIKE '%REP%';

4 rows inserted
```

- N'utilisez pas la clause `VALUES`.
- Le nombre de colonnes de la clause `INSERT` doit correspondre à celui de la sous-interrogation.
- Insérez toutes les lignes renvoyées par la sous-interrogation dans la table `sales_reps`.

Modifier des données dans une table

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	60
104	Bruce	Ernst	6000	103	(null)	60
107	Diana	Lorentz	4200	103	(null)	60
124	Kevin	Mourgos	5800	100	(null)	50

Mettre à jour les lignes de la table EMPLOYEES :

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	MANAGER_ID	COMMISSION_PCT	DEPARTMENT_ID
100	Steven	King	24000	(null)	(null)	90
101	Neena	Kochhar	17000	100	(null)	90
102	Lex	De Haan	17000	100	(null)	90
103	Alexander	Hunold	9000	102	(null)	80
104	Bruce	Ernst	6000	103	(null)	80
107	Diana	Lorentz	4200	103	(null)	80
124	Kevin	Mourgos	5800	100	(null)	50

Syntaxe de l'instruction `UPDATE`

- Modifiez les valeurs existantes d'une table avec l'instruction `UPDATE` :

```
UPDATE    table
SET    column = value [, column = value, ...]
[WHERE       condition];
```

- Mettez à jour plusieurs lignes à la fois (si nécessaire).

Mettre à jour des lignes d'une table

- Si vous indiquez la clause `WHERE`, seules les valeurs d'une ou plusieurs lignes spécifiques sont modifiées :

```
UPDATE employees
SET    department_id = 50
WHERE  employee_id = 113;
1 rows updated
```

- Si vous omettez la clause `WHERE`, les valeurs de toutes les lignes de la table sont modifiées :

```
UPDATE     copy_emp
SET        department_id = 110;
22 rows updated
```

- Indiquez `SET` *`column_name`*`= NULL` pour remplacer la valeur d'une colonne par `NULL`.

Mettre à jour deux colonnes avec une sous-interrogation

Mettez à jour le poste et le salaire de l'employé 113 de sorte qu'ils correspondent à ceux de l'employé 205.

```
UPDATE    employees
SET       job_id   = (SELECT   job_id
                      FROM     employees
                      WHERE    employee_id = 205),
          salary   = (SELECT   salary
                      FROM     employees
                      WHERE    employee_id = 205)
WHERE     employee_id     =  113;
```

```
1 rows updated
```

Mettre à jour des lignes sur la base d'une autre table

Utilisez les sous-interrogations des instructions `UPDATE` pour mettre à jour les valeurs des lignes d'une table sur la base des valeurs d'une autre table :

```
UPDATE   copy_emp
SET      department_id  =  (SELECT department_id
                           FROM employees
                           WHERE employee_id = 100)
WHERE    job_id         =  (SELECT job_id
                           FROM employees
                           WHERE employee_id = 200);
1 rows updated
```

Supprimer une ligne d'une table

DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700
8	190	Contracting	(null)	1700

Supprimer une ligne de la table DEPARTMENTS :

	DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
1	10	Administration	200	1700
2	20	Marketing	201	1800
3	50	Shipping	124	1500
4	60	IT	103	1400
5	80	Sales	149	2500
6	90	Executive	100	1700
7	110	Accounting	205	1700

Instruction DELETE

Vous pouvez supprimer des lignes existantes d'une table à l'aide de l'instruction DELETE :

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

Supprimer des lignes d'une table

- Si vous indiquez la clause `WHERE`, des lignes spécifiques sont supprimées :

```
DELETE FROM departments
WHERE  department_name = 'Finance';
1 rows deleted
```

- Si vous omettez la clause `WHERE`, toutes les lignes de la table sont supprimées :

```
DELETE FROM  copy_emp;
22 rows deleted
```

Supprimer des lignes sur la base d'une autre table

Utilisez les sous-interrogations des instructions `DELETE` pour supprimer des lignes d'une table sur la base des valeurs d'une autre table :

```
DELETE FROM employees
WHERE  department_id =
                (SELECT department_id
                 FROM   departments
                 WHERE  department_name
                        LIKE '%Public%');
```

```
1 rows deleted
```

Instruction TRUNCATE

- Elle supprime toutes les lignes d'une table, en laissant cette dernière vide sans toucher à sa structure.
- Il s'agit d'une instruction LDD (langage de définition de données) et non d'une instruction LMD. Il est donc difficile de l'annuler.
- Syntaxe :

```
TRUNCATE TABLE table_name;
```

- Exemple :

```
TRUNCATE TABLE copy_emp;
```

Transactions de base de données

Une transaction de base de données se compose des éléments suivants :

- Des instructions LMD qui apportent une modification cohérente
  aux données
- Une instruction LDD
- Une instruction LCD (langage de contrôle de données)

Transactions de base de données : Début et fin

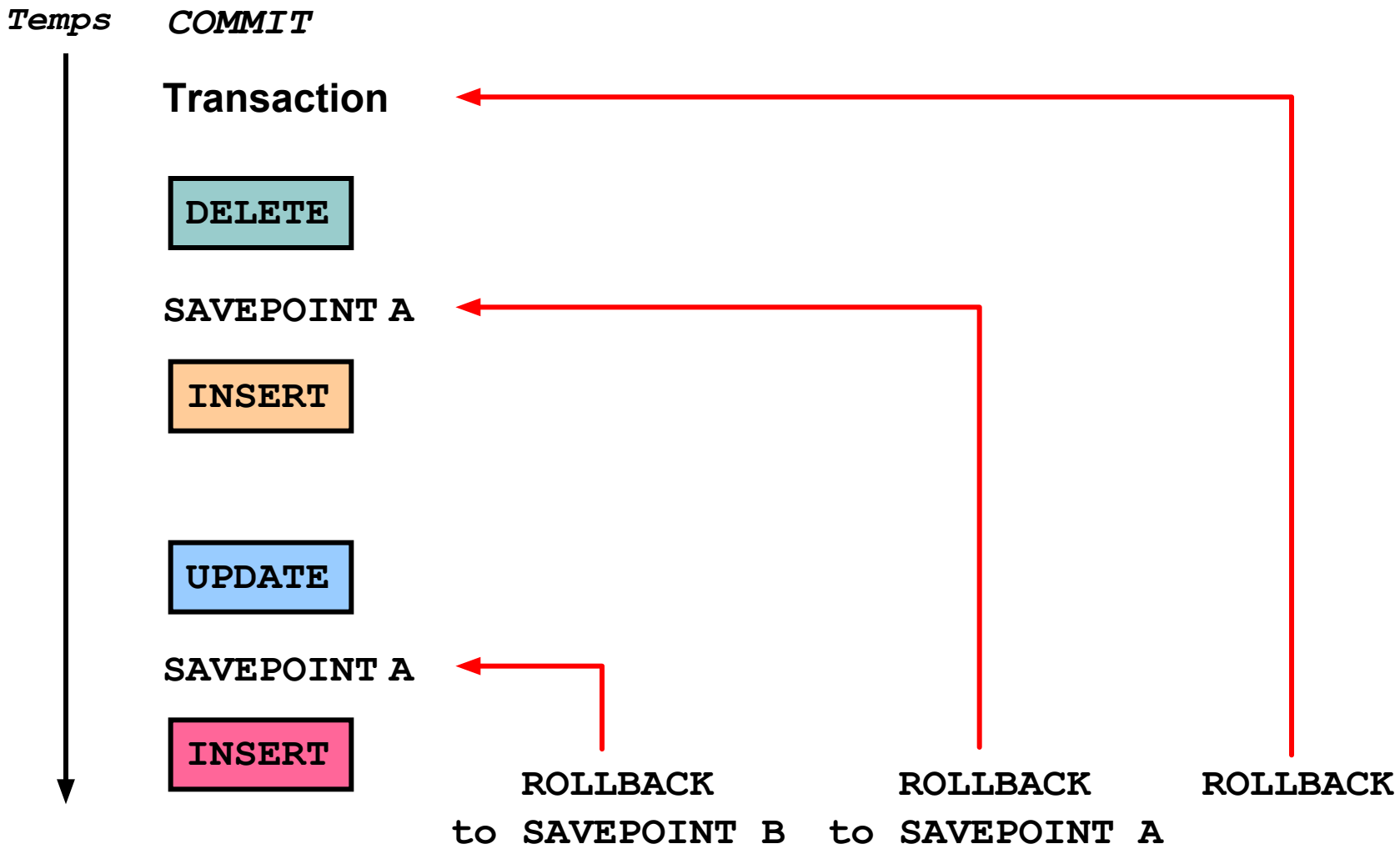
- Elles commencent avec l'exécution de la première instruction SQL LMD.
- Elles finissent lorsque l'un des événements suivants se produit :
  - Une instruction `COMMIT` ou `ROLLBACK` est exécutée.
  - Une instruction LDD ou LCD est exécutée (validation automatique).
  - L'utilisateur quitte SQL Developer ou SQL*Plus.
  - Le système connaît une défaillance.

Avantages des instructions COMMIT et ROLLBACK

Avec les instructions COMMIT et ROLLBACK, vous pouvez :

- garantir la cohérence des données
- prévisualiser les modifications apportées aux données avant de les rendre définitives
- regrouper de façon logique les opérations associées

Instructions explicites de contrôle des transactions



Annuler des modifications jusqu'à un marqueur

- Créez un marqueur dans la transaction en cours à l'aide de l'instruction `SAVEPOINT`.
- Procédez à une annulation jusqu'à ce marqueur à l'aide de l'instruction `ROLLBACK TO SAVEPOINT`.

```
UPDATE...
SAVEPOINT update_done;
SAVEPOINT update_done succeeded.
INSERT...
ROLLBACK TO update_done;
ROLLBACK TO succeeded.
```

Traitement implicite des transactions

- Une validation automatique a lieu dans les cas suivants :
  - Une instruction LDD est exécutée.
  - Une instruction LCD est exécutée.
  - SQL Developer ou SQL*Plus est fermé normalement, sans exécution explicite d'instructions `COMMIT` or `ROLLBACK`.
- Une annulation automatique se produit en cas d'arrêt anormal de SQL Developer ou de SQL*Plus, ou de défaillance du système.

Etat des données avant exécution de l'instruction `COMMIT` ou `ROLLBACK`

- L'état antérieur des données peut être récupéré.
- L'utilisateur actuel peut visualiser les résultats des opérations LMD à l'aide de l'instruction `SELECT`.
- Les autres utilisateurs *ne peuvent pas* afficher les résultats des instructions LMD exécutées par l'utilisateur actuel.
- Les lignes affectées sont *verrouillées*. Les autres utilisateurs ne peuvent donc pas modifier les données de ces lignes.

Etat des données après exécution de l'instruction `COMMIT`

- Les modifications apportées aux données sont enregistrées dans la base.
- L'état antérieur des données est écrasé.
- Tous les utilisateurs peuvent visualiser les résultats.
- Les verrous externes des lignes affectées sont libérés. Ces lignes peuvent alors être manipulées par les autres utilisateurs.
- Tous les savepoints sont effacés.

Valider des données

- Apportez des modifications :

```
DELETE FROM employees
WHERE  employee_id = 99999;
1 rows deleted

INSERT INTO departments
VALUES (290, 'Corporate Tax', NULL, 1700);
1 rows inserted
```

- Validez les modifications :

```
COMMIT;
COMMIT succeeded.
```

Etat des données après exécution de l'instruction `ROLLBACK`

Annulez toutes les modifications en attente à l'aide de l'instruction `ROLLBACK` :

- Les modifications apportées aux données sont annulées.
- L'état antérieur des données est restauré.
- Les verrous externes des lignes affectées sont libérés.

```
DELETE FROM copy_emp;
ROLLBACK ;
```

Etat des données après exécution de l'instruction `ROLLBACK` : Exemple

```
DELETE FROM test;
25,000 rows deleted.

ROLLBACK;
Rollback complete.

DELETE FROM test WHERE  id = 100;
1 row deleted.

SELECT * FROM   test WHERE  id = 100;
No rows selected.

COMMIT;
Commit complete.
```

Annulation au niveau instruction

- Si une instruction LMD unique échoue lors de l'exécution, seule cette instruction est annulée.
- Le serveur Oracle implémente un savepoint implicite.
- Toutes les autres modifications sont conservées.
- L'utilisateur doit terminer les transactions de façon explicite à l'aide d'une instruction `COMMIT` ou `ROLLBACK`.

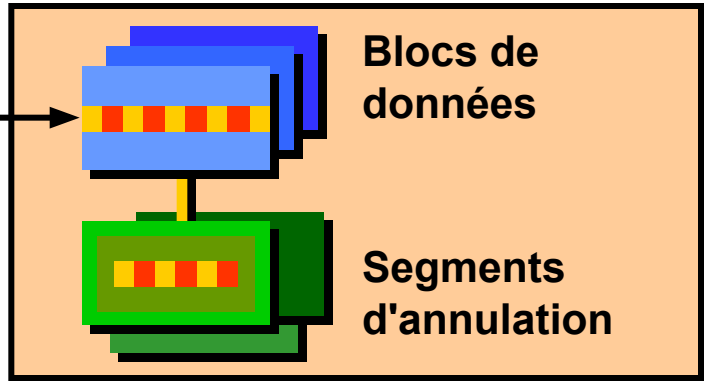
Cohérence en lecture

- La cohérence en lecture garantit une vue cohérente des données à tout moment.
- Les modifications apportées par un utilisateur n'entrent pas en conflit avec les modifications apportées par un autre utilisateur.
- La cohérence en lecture garantit que, pour les mêmes données :
  - Les utilisateurs qui lisent n'attendent pas que ceux qui écrivent aient terminé.
  - Les utilisateurs qui écrivent n'attendent pas que ceux qui lisent aient terminé.
  - Les utilisateurs qui écrivent doivent attendre que les autres utilisateurs qui écrivent aient terminé.

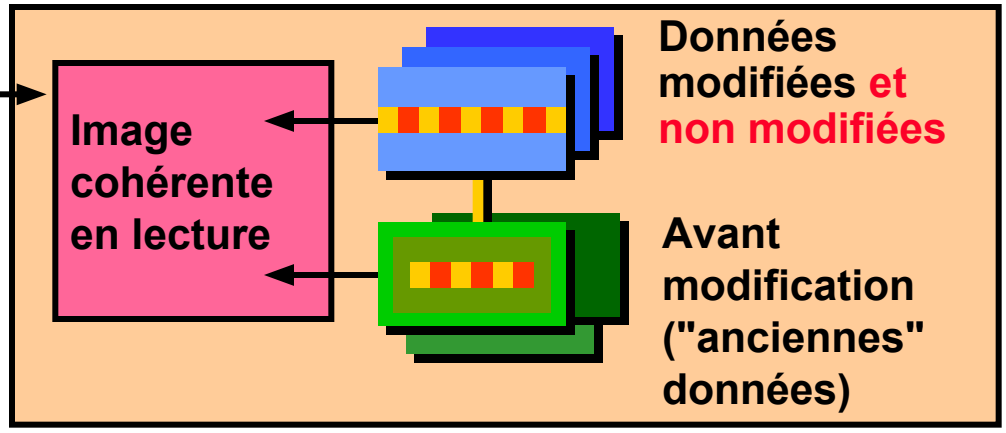
Implémenter la cohérence en lecture

Utilisateur A

```
UPDATE employees
SET    salary = 7000
WHERE  last_name = 'Grant';
```



```
SELECT  *
FROM userA.employees;
```



Utilisateur B

Clause FOR UPDATE dans une instruction SELECT

- Dans la table EMPLOYEES, verrouillez les lignes pour lesquelles job_id = SA_REP.

```
SELECT employee_id, salary, commission_pct, job_id
FROM employees
WHERE job_id = 'SA_REP'
FOR UPDATE
ORDER BY employee_id;
```

- Le verrou est libéré uniquement lorsque vous exécutez une instruction ROLLBACK ou COMMIT.
- Si l'instruction SELECT tente de verrouiller une ligne qui est déjà verrouillée par un autre utilisateur, la base de données attend que la ligne soit disponible pour renvoyer les résultats de l'instruction SELECT.

Clause FOR UPDATE : Exemples

- Dans une instruction SELECT, vous pouvez utiliser la clause FOR UPDATE sur plusieurs tables.

```
SELECT e.employee_id, e.salary, e.commission_pct
FROM employees e JOIN departments d
USING (department_id)
WHERE job_id = 'ST_CLERK'
AND location_id = 1500
FOR UPDATE
ORDER BY e.employee_id;
```

- Des lignes des tables EMPLOYEES et DEPARTMENTS sont verrouillées.
- Utilisez FOR UPDATE OF *column_name* pour qualifier la colonne que vous avez l'intention de modifier. Seules les lignes de cette table spécifique sont alors verrouillées.