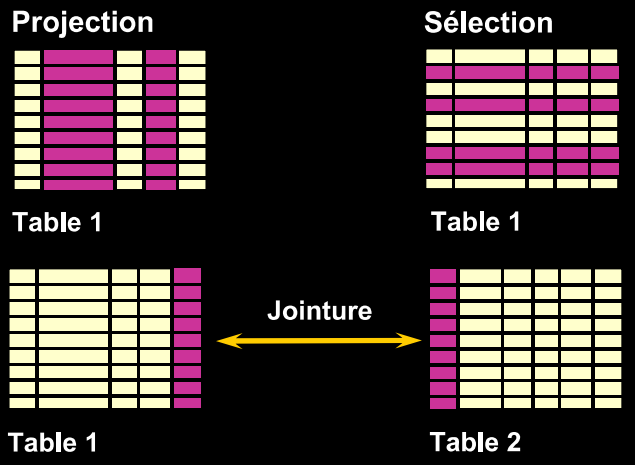
1

Ecrire des instructions SQL SELECT élémentaires

Objectifs

- énumérer toutes les possibilités offertes par les instructions SQL `SELECT`
- exécuter une instruction `SELECT` élémentaire

Différentes fonctions des instructions SQL `SELECT`



Instruction `SELECT` élémentaire

```
SELECT   *|{[DISTINCT] column|expression [alias],...}
FROM     table;
```

- `SELECT` indique *quelles* colonnes renvoyer
- `FROM` indique *dans quelle* table rechercher

Sélectionner toutes les colonnes

```
SELECT *
FROM   departments;
```

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500
90	Executive	100	1700
110	Accounting	205	1700
190	Contracting		1700

8 rows selected.

Sélectionner des colonnes spécifiques

```
SELECT department_id, location_id
FROM   departments;
```

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

Utiliser des opérateurs arithmétiques

```
SELECT last_name, salary, salary + 300
FROM   employees;
```

LAST_NAME	SALARY	SALARY+300
King	24000	24300
Kochhar	17000	17300
De Haan	17000	17300
Hunold	9000	9300
Ernst	6000	6300
...		
Hartstein	13000	13300
Fay	6000	6300
Higgins	12000	12300
Gietz	8300	8600

20 rows selected.

Priorité des opérateurs

```
SELECT last_name, salary, 12*salary+100
FROM   employees;
```

LAST_NAME	SALARY	12*SALARY+100
King	24000	288100
Kochhar	17000	204100
De Haan	17000	204100
Hunold	9000	108100
Ernst	6000	72100
...		
Hartstein	13000	156100
Fay	6000	72100
Higgins	12000	144100
Gietz	8300	99700

20 rows selected.

Utiliser des parenthèses

```
SELECT last_name, salary, 12*(salary+100)
FROM   employees;
```

LAST_NAME	SALARY	12*(SALARY+100)
King	24000	289200
Kochhar	17000	205200
De Haan	17000	205200
Hunold	9000	109200
Ernst	6000	73200
...		
Hartstein	13000	157200
Fay	6000	73200
Higgins	12000	145200
Gietz	8300	100800

20 rows selected.

Définir une valeur NULL

- Une valeur NULL est une valeur non disponible, non affectée, inconnue ou inapplicable.
- La valeur NULL est différente du zéro ou de l'espace.

```
SELECT last_name, job_id, salary, commission_pct
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	COMMISSION_PCT
King	AD_PRES	24000	
Kochhar	AD_VP	17000	
...			
Zlotkey	SA_MAN	10500	.2
Abel	SA_REP	11000	.3
Taylor	SA_REP	8600	.2
...			
Gietz	AC_ACCOUNT	8300	

20 rows selected.

Valeurs NULL dans les expressions arithmétiques

Les expressions arithmétiques comportant une valeur NULL ont pour résultat une valeur NULL.

```
SELECT last_name, 12*salary*commission_pct
FROM   employees;
```

LAST_NAME	12*SALARY*COMMISSION_PCT
King	
Kochhar	
...	
Zlotkey	25200
Abel	39600
Taylor	20640
...	
Gietz	

20 rows selected.

Définir un alias de colonne

L'alias de colonne :

- renomme un en-tête de colonne,
- est utile dans les calculs,
- suit le nom de la colonne (le mot-clé `AS` facultatif peut être placé entre le nom de la colonne et l'alias),
- doit obligatoirement être placé entre guillemets s'il contient des espaces ou des caractères spéciaux, ou bien si les majuscules/minuscules doivent être respectées.

Utiliser des alias de colonne

```
SELECT last_name AS name, commission_pct comm
FROM   employees;
```

NAME	COMM
King	
Kochhar	
De Haan	

...

20 rows selected.

```
SELECT last_name "Name", salary*12 "Annual Salary"
FROM   employees;
```

Name	Annual Salary
King	288000
Kochhar	204000
De Haan	204000

...

20 rows selected.

Opérateur de concaténation

Un opérateur de concaténation :

- concatène des colonnes ou des chaînes de caractères avec d'autres colonnes,
- est représenté par deux barres verticales (||),
- crée une colonne qui contient une expression alphanumérique.

Utiliser l'opérateur de concaténation

```
SELECT   last_name||job_id AS "Employees"
FROM     employees;
```

Employees
KingAD_PRES
KochharAD_VP
De HaanAD_VP
HunoldIT_PROG
ErnstIT_PROG
LorentzIT_PROG
MourgosST_MAN
RajsST_CLERK

...

20 rows selected.

Utiliser des chaînes de caractères

```
SELECT last_name  ||' is a '||job_id
       AS "Employee Details"
FROM   employees;
```

Employee Details
King is a AD_PRES
Kochhar is a AD_VP
De Haan is a AD_VP
Hunold is a IT_PROG
Ernst is a IT_PROG
Lorentz is a IT_PROG
Mourgos is a ST_MAN
Rajs is a ST_CLERK

...

20 rows selected.

Doublons

Par défaut, le résultat d'une interrogation affiche toutes les lignes, y compris les doublons.

```
SELECT department_id
FROM   employees;
```

DEPARTMENT_ID
90
90
90
60
60
60
50
50
50

...

20 rows selected.

Eliminer les doublons

Pour éliminer les doublons, ajoutez le mot-clé `DISTINCT` dans la clause `SELECT`.

```
SELECT DISTINCT department_id
FROM   employees;
```

DEPARTMENT_ID
10
20
50
60
80
90
110

8 rows selected.

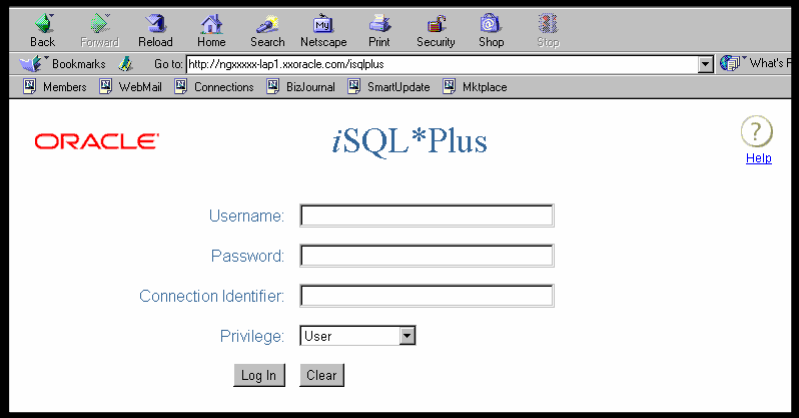
Présentation d'*i*SQL*Plus

Une fois que vous êtes connecté à *i*SQL*Plus, vous pouvez :

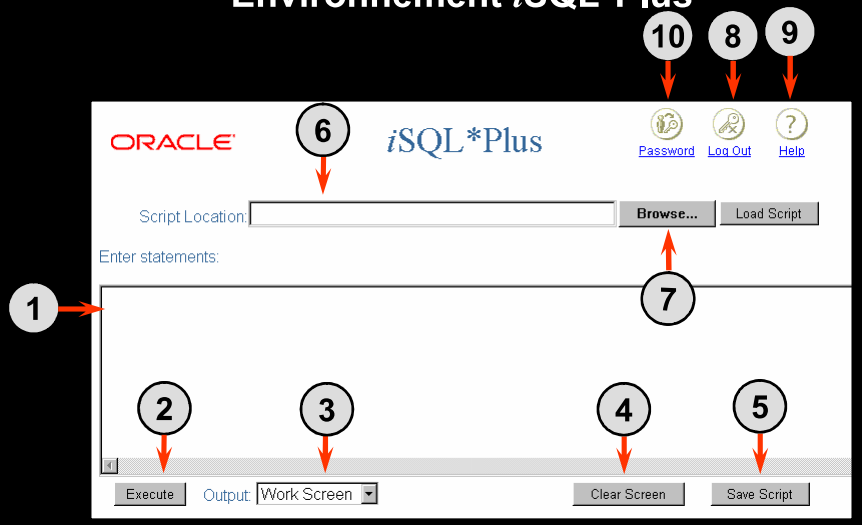
- décrire la structure d'une table,
- éditer une instruction SQL,
- exécuter SQL,
- enregistrer et ajouter des instructions SQL dans des fichiers,
- exécuter des instructions stockées dans des fichiers sauvegardés,
- charger des commandes depuis un fichier texte dans la fenêtre d'édition d'*i*SQL*Plus.

Se connecter à *i*SQL*Plus

Depuis l'environnement de votre navigateur Windows :



Environnement *i*SQL*Plus



Afficher la structure d'une table

Utilisez la commande `DESCRIBE` d'*i*SQL*Plus pour afficher la structure d'une table .

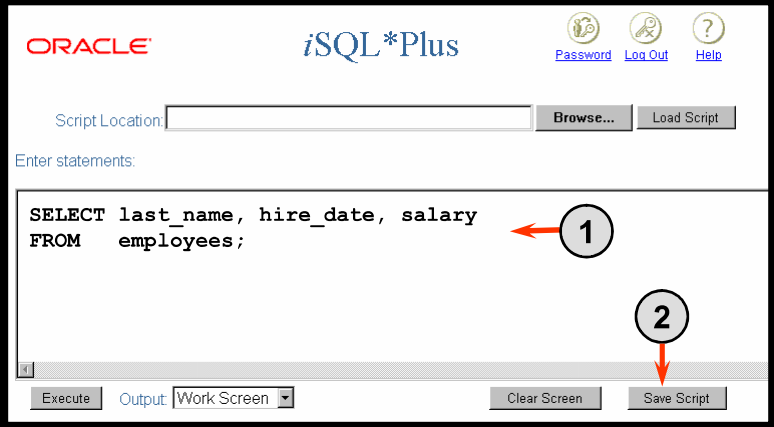
```
DESC[RIBE] tablename
```


Afficher la structure d'une table

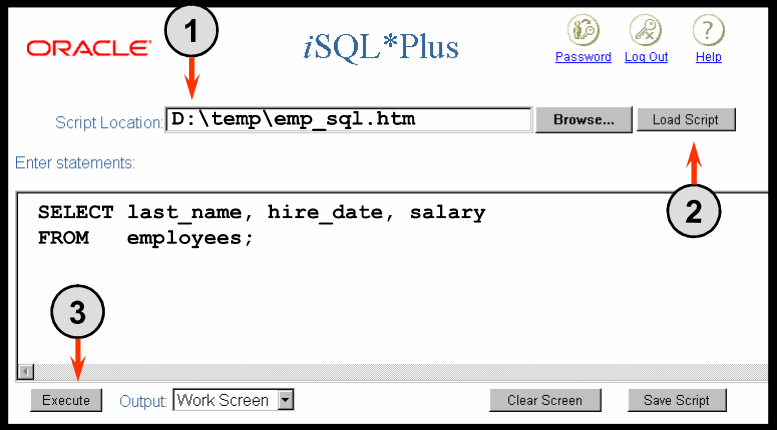
```
DESCRIBE employees
```

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)

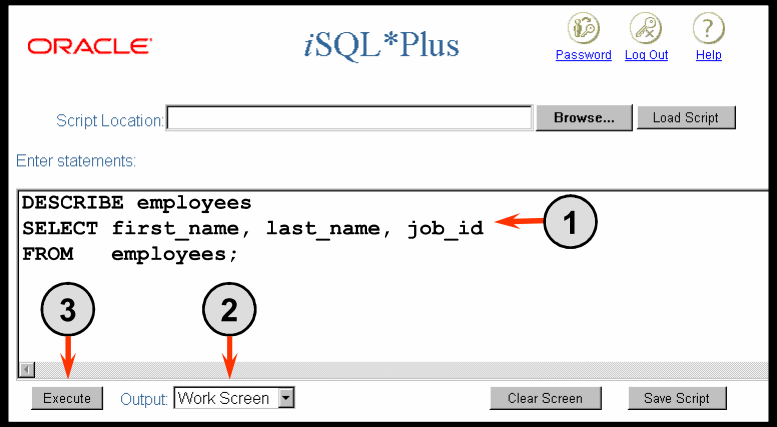
Interagir avec des fichiers script



Interagir avec des fichiers script



Interagir avec des fichiers script



Présentation de l'exercice 1

Dans cet exercice, vous allez :

- sélectionner l'ensemble des données de différentes tables
- afficher la structure des tables
- effectuer des calculs arithmétiques et indiquer des noms de colonne
- utiliser *i*SQL*Plus

2

Limiter et trier des données

ORACLE

Objectifs

- limiter le nombre de lignes extraites par une interrogation
- trier les lignes extraites par une interrogation

Limiter le nombre de lignes à l'aide d'une sélection

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90
103	Hunold	IT_PROG	60
104	Ernst	IT_PROG	60
107	Lorentz	IT_PROG	60
124	Mourgos	ST_MAN	50

...

20 rows selected.

"Extraire tous les employés du service 90"

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

Limiter le nombre de lignes sélectionnées

- Limitez le nombre de lignes renvoyées à l'aide de la clause `WHERE`.

```
SELECT    *|{[DISTINCT] column|expression [alias],...}
FROM      table
[WHERE    condition(s)];
```

- La clause `WHERE` se place après la clause `FROM`.

Utiliser la clause WHERE

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees
WHERE  department_id = 90 ;
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90

Chaînes de caractères et dates

- Les chaînes de caractères et les dates doivent être placées entre apostrophes.
- La recherche tient compte des majuscules/minuscules pour les chaînes de caractères et du format pour les dates.
- Le format de date par défaut est DD-MON-RR.

```
SELECT last_name, job_id, department_id
FROM   employees
WHERE  last_name = 'Whalen';
```


Conditions de comparaison

Opérateur	Signification
=	Egal à
>	Supérieur à
>=	Supérieur ou égal à
<	Inférieur à
<=	Inférieur ou égal à
<>	Différent de

Autres conditions de comparaison

Opérateur	Signification
BETWEEN ...AND...	Compris entre ... et ... (bornes comprises)
IN(set)	Correspond à une valeur de la liste
LIKE	Ressemblance partielle de chaînes de caractères
IS NULL	Correspond à une valeur NULL

Utiliser la condition BETWEEN

Utilisez la condition BETWEEN pour afficher des lignes en fonction d'une plage de valeurs.

```
SELECT last_name, salary
FROM   employees
WHERE  salary BETWEEN 2500 AND 3500;
```

Limite inférieure Limite supérieure

LAST_NAME	SALARY
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500

Utiliser la condition IN

Utilisez la condition d'appartenance IN pour vérifier la présence de valeurs dans une liste.

```
SELECT employee_id, last_name, salary, manager_id
FROM   employees
WHERE  manager_id IN (100, 101, 201);
```

EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
202	Fay	6000	201
200	Whalen	4400	101
205	Higgins	12000	101
101	Kochhar	17000	100
102	De Haan	17000	100
124	Mourgos	5800	100
149	Zlotkey	10500	100
201	Hartstein	13000	100

8 rows selected.

Utiliser la condition LIKE

- Utilisez la condition LIKE pour rechercher des chaînes de caractères valides à l'aide de caractères génériques.
- Les conditions de recherche peuvent contenir des caractères ou des nombres littéraux :
  - % représente zéro ou plusieurs caractères.
  - _ représente un caractère.

```
SELECT   first_name
FROM     employees
WHERE    first_name LIKE 'S%';
```


Utiliser la condition LIKE

- Vous pouvez combiner plusieurs caractères génériques de recherche.

```
SELECT last_name
FROM   employees
WHERE  last_name LIKE '_o%';
```

LAST_NAME
Kochhar
Lorentz
Mourgos

- Vous pouvez utiliser l'identificateur ESCAPE pour rechercher les symboles % et _.

Utiliser les conditions NULL

Recherchez des valeurs NULL avec l'opérateur IS NULL.

```
SELECT last_name, manager_id
FROM   employees
WHERE  manager_id IS NULL;
```

LAST_NAME	MANAGER_ID
King	

Conditions logiques

Opérateur	Signification
AND	Renvoie TRUE si les *deux* conditions sont vraies
OR	Renvoie TRUE si *l'une* des conditions est vraie
NOT	Renvoie la valeur TRUE si la condition qui suit l'opérateur est fausse

Utiliser l'opérateur AND

L'opérateur AND exige que les deux conditions soient vraies.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >=10000
AND    job_id LIKE '%MAN%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
149	Zlotkey	SA_MAN	10500
201	Hartstein	MK_MAN	13000

Utiliser l'opérateur OR

L'opérateur OR exige que l'une des conditions soit vraie.

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary >= 10000
OR     job_id LIKE '%MAN%';
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
100	King	AD_PRES	24000
101	Kochhar	AD_VP	17000
102	De Haan	AD_VP	17000
124	Mourgos	ST_MAN	5800
149	Zlotkey	SA_MAN	10500
174	Abel	SA_REP	11000
201	Hartstein	MK_MAN	13000
205	Higgins	AC_MGR	12000

8 rows selected.

Utiliser l'opérateur NOT

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id
       NOT IN ('IT_PROG', 'ST_CLERK', 'SA_REP');
```

LAST_NAME	JOB_ID
King	AD_PRES
Kochhar	AD_VP
De Haan	AD_VP
Mourgos	ST_MAN
Zlotkey	SA_MAN
Whalen	AD_ASST
Hartstein	MK_MAN
Fay	MK_REP
Higgins	AC_MGR
Gietz	AC_ACCOUNT

10 rows selected.

Règles de priorité

Ordre d'évaluation	Opérateur
1	Opérateurs arithmétiques
2	Opérateur de concaténation
3	Conditions de comparaison
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Condition logique NOT
7	Condition logique AND
8	Condition logique OR

Les parenthèses permettent de modifier les règles de priorité.

Règles de priorité

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = 'SA_REP'
OR     job_id = 'AD_PRES'
AND    salary > 15000;
```

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000
Abel	SA_REP	11000
Taylor	SA_REP	8600
Grant	SA_REP	7000

Règles de priorité

Utilisez des parenthèses pour forcer la priorité.

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  (job_id = 'SA_REP'
OR     job_id = 'AD_PRES')
AND    salary > 15000;
```

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000

Clause ORDER BY

- Triez des lignes à l'aide de la clause ORDER BY.
  - ASC : ordre croissant (par défaut)
  - DESC : ordre décroissant
- La clause ORDER BY se place à la fin de l'instruction SELECT.

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date ;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
King	AD_PRES	90	17-JUN-87
Whalen	AD_ASST	10	17-SEP-87
Kochhar	AD_VP	90	21-SEP-89
Hunold	IT_PROG	60	03-JAN-90
Ernst	IT_PROG	60	21-MAY-91

...

20 rows selected.

Trier par ordre décroissant

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY hire_date DESC ;
```

LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
Zlotkey	SA_MAN	80	29-JAN-00
Mourgos	ST_MAN	50	16-NOV-99
Grant	SA_REP		24-MAY-99
Lorentz	IT_PROG	60	07-FEB-99
Vargas	ST_CLERK	50	09-JUL-98
Taylor	SA_REP	80	24-MAR-98
Matos	ST_CLERK	50	15-MAR-98
Fay	MK_REP	20	17-AUG-97
Davies	ST_CLERK	50	29-JAN-97

...

20 rows selected.

Trier par alias de colonne

```
SELECT employee_id, last_name, salary*12 annsal
FROM   employees
ORDER BY annsal;
```

EMPLOYEE_ID	LAST_NAME	ANNSAL
144	Vargas	30000
143	Matos	31200
142	Davies	37200
141	Rajs	42000
107	Lorentz	50400
200	Whalen	52800
124	Mourgos	69600
104	Ernst	72000
202	Fay	72000
178	Grant	84000

...

20 rows selected.

Trier sur plusieurs colonnes

- L'ordre des éléments de la liste ORDER BY donne l'ordre du tri.

```
SELECT last_name, department_id, salary
FROM   employees
ORDER BY department_id, salary DESC;
```

LAST_NAME	DEPARTMENT_ID	SALARY
Whalen	10	4400
Hartstein	20	13000
Fay	20	6000
Mourgos	50	5800
Rajs	50	3500
Davies	50	3100
Matos	50	2600
Vargas	50	2500

...

20 rows selected.

- Vous pouvez effectuer un tri sur une colonne ne figurant pas dans la liste SELECT.

Synthèse

- utiliser la clause `WHERE` pour limiter le nombre de lignes de résultat
  - utiliser les conditions de comparaison
  - utiliser les conditions `BETWEEN`, `IN`, `LIKE` et `NULL`
  - appliquer les opérateurs logiques `AND`, `OR` et `NOT`
- utiliser la clause `ORDER BY` pour trier les lignes de résultat

```
SELECT      *|{[DISTINCT] column|expression [alias],...}
FROM        table
[WHERE      condition(s)]
[ORDER BY   {column, expr, alias} [ASC|DESC]];
```

ORACLE

Présentation de l'exercice 2

Dans cet exercice, vous allez :

- sélectionner des données et modifier l'ordre d'affichage des lignes,
- limiter le nombre de lignes à l'aide de la clause `WHERE`,
- trier des lignes à l'aide de la clause `ORDER BY`.

ORACLE

ORACLE

3

Afficher des données issues de plusieurs tables

ORACLE

Objectifs

- écrire des instructions `SELECT` pour accéder aux données de plusieurs tables en utilisant des équijointures et des non-équijointures
- visualiser des données ne répondant pas aux conditions de jointure en utilisant des jointures externes
- joindre une table à elle-même à l'aide d'une auto-jointure

Afficher des données issues de plusieurs tables

EMPLOYEES

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700

EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
200	10	Administration
201	20	Marketing
202	20	Marketing
...		
102	90	Executive
205	110	Accounting
206	110	Accounting

Produits cartésiens

- Un produit cartésien est généré :
  - lorsqu'une condition de jointure est omise,
  - lorsqu'une condition de jointure est incorrecte,
  - lorsque toutes les lignes de la première table sont jointes à toutes les lignes de la seconde.
- Pour éviter tout produit cartésien, insérez une condition de jointure correcte dans la clause `WHERE`.

Générer un produit cartésien

EMPLOYEES (20 lignes)

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
...		
202	Fay	20
205	Higgins	110
206	Gietz	110

20 rows selected.

DEPARTMENTS (8 lignes)

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
10	Administration	1700
20	Marketing	1800
50	Shipping	1500
60	IT	1400
80	Sales	2500
90	Executive	1700
110	Accounting	1700
190	Contracting	1700

8 rows selected.

Produit cartésien : 20x8=160 lignes

EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
100	90	1700
101	90	1700
102	90	1700
103	60	1700
104	60	1700
107	60	1700
...		

160 rows selected.

Types de jointure

Jointures conformes à la norme SQL: 1999 :

- Jointures croisées
- Jointures naturelles
- Clause Using
- Jointures externes complètes, également appelées jointures externe gauche et droite
- Conditions de jointure arbitraires pour jointures externes

Joindre des tables à l'aide de la syntaxe Oracle

Une jointure sert à interroger des données de plusieurs tables.

```
SELECT   table1.column, table2.column
FROM     table1, table2
WHERE    table1.column1 = table2.column2;
```

- Ecrivez la condition de jointure dans la clause **WHERE**.
- Placez le nom de la table avant le nom de la colonne lorsque celui-ci figure dans plusieurs tables.

Définition d'une équijointure

EMPLOYEES

EMPLOYEE_ID	DEPARTMENT_ID
200	10
201	20
202	20
124	50
141	50
142	50
143	50
144	50
103	60
104	60
107	60
149	80
174	80
176	80

...

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping
60	IT
60	IT
60	IT
80	Sales
80	Sales
80	Sales

...

Clé étrangère Clé primaire

Extraire des enregistrements à l'aide d'équijointures

```
SELECT employees.employee_id, employees.last_name,
       employees.department_id, departments.department_id,
       departments.location_id
FROM   employees, departments
WHERE  employees.department_id = departments.department_id;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800
124	Mourgos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500
143	Matos	50	50	1500
144	Vargas	50	50	1500

...

19 rows selected.

Autres conditions de recherche utilisant l'opérateur AND

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
Whalen	10
Hartstein	20
Fay	20
Mourgos	50
Rajs	50
Davies	50
Matos	50
Vargas	50
Hunold	60
Ernst	60

...

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping
60	IT
60	IT

...

Différencier les noms de colonne

- Utilisez des préfixes qui précisent le nom de la table pour différencier les noms de colonne appartenant à plusieurs tables.
- L'utilisation de préfixes désignant la table améliore les performances.
- Différenciez des colonnes de même nom appartenant à plusieurs tables en utilisant des alias de colonne.

Utiliser des alias de table

- Simplifiez les interrogations à l'aide des alias de table.
- L'utilisation de préfixes désignant la table améliore les performances.

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e , departments d
WHERE  e.department_id = d.department_id;
```


Joindre plus de deux tables

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
King	90
Kochhar	90
De Haan	90
Hunold	60
Ernst	60
Lorentz	60
Mourgos	50
Rajs	50
Davies	50
Matos	50
Vargas	50
Zlotkey	80
Abel	80
Taylor	80

...

20 rows selected.

DEPARTMENTS

DEPARTMENT_ID	LOCATION_ID
10	1700
20	1800
50	1500
60	1400
80	2500
90	1700
110	1700
190	1700

8 rows selected.

LOCATIONS

LOCATION_ID	CITY
1400	Southlake
1500	South San Francisco
1700	Seattle
1800	Toronto
2500	Oxford

Pour joindre *n* tables entre elles, il faut au minimum n-1 conditions de jointure. Par exemple, deux jointures au moins sont nécessaires pour joindre trois tables.

Non-équijointures

EMPLOYEES

LAST_NAME	SALARY
King	24000
Kochhar	17000
De Haan	17000
Hunold	9000
Ernst	6000
Lorentz	4200
Mourgos	5800
Rajs	3500
Davies	3100
Matos	2600
Vargas	2500
Zlotkey	10500
Abel	11000
Taylor	8600

...

20 rows selected.

JOB_GRADES

GRA	LOWEST_SAL	HIGHEST_SAL
A	1000	2999
B	3000	5999
C	6000	9999
D	10000	14999
E	15000	24999
F	25000	40000

Les salaires de la table EMPLOYEES doivent être compris entre le salaire minimal et le salaire maximal de la table JOB_GRADES.

Extraire des enregistrements à l'aide de non-équijointures

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e, job_grades j
WHERE  e.salary
       BETWEEN j.lowest_sal AND j.highest_sal;
```

LAST_NAME	SALARY	GRA
Matos	2600	A
Vargas	2500	A
Lorentz	4200	B
Mourgos	5800	B
Rajs	3500	B
Davies	3100	B
Whalen	4400	B
Hunold	9000	C
Ernst	6000	C

...

20 rows selected.

Jointures externes

DEPARTMENTS

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

8 rows selected.

EMPLOYEES

DEPARTMENT_ID	LAST_NAME
90	King
90	Kochhar
90	De Haan
60	Hunold
60	Ernst
60	Lorentz
50	Mourgos
50	Rajs
50	Davies
50	Matos
50	Vargas
80	Zlotkey

...

20 rows selected.

Le service 190 ne comprend pas d'employés.

Syntaxe des jointures externes

- Les jointures externes permettent de visualiser également des lignes qui ne répondent pas à la condition de jointure.
- L'opérateur de jointure externe est le signe (+).

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column(+) = table2.column;
```

```
SELECT table1.column, table2.column
FROM   table1, table2
WHERE  table1.column = table2.column(+);
```

Utiliser des jointures externes

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e, departments d
WHERE  e.department_id(+) = d.department_id ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Mourgos	50	Shipping
Rajs	50	Shipping
Davies	50	Shipping
Matos	50	Shipping
...		
Gietz	110	Accounting
		Contracting

20 rows selected.

Auto-jointures

EMPLOYEES (WORKER)

EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100
...		

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
124	Mourgos
...	

Dans la table WORKER, MANAGER_ID équivaut à EMPLOYEE_ID dans la table MANAGER.

Joindre une table à elle-même

```
SELECT worker.last_name || ' works for '
       || manager.last_name
FROM   employees worker, employees manager
WHERE  worker.manager_id = manager.employee_id ;
```

WORKER.LAST_NAME||'WORKSFOR'|MANAGER.LAST_NAME
Kochhar works for King
De Haan works for King
Mourgos works for King
Zlotkey works for King
Hartstein works for King
Whalen works for Kochhar
Higgins works for Kochhar
Hunold works for De Haan
Ernst works for Hunold
...

19 rows selected.

Présentation de l'exercice 3

Dans cet exercice, vous allez écrire des instructions pour joindre des tables à l'aide de la syntaxe Oracle.

Joindre des tables à l'aide de la syntaxe SQL: 1999

Une jointure permet d'interroger des données de plusieurs tables.

```
SELECT   table1.column, table2.column
FROM     table1
[CROSS JOIN table2] |
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON(table1.column_name = table2.column_name)] |
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)];
```


Créer des jointures croisées

- La clause `CROSS JOIN` génère le produit cartésien de deux tables.

```
SELECT last_name, department_name
FROM   employees
CROSS JOIN departments ;
```

LAST_NAME	DEPARTMENT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
Hunold	Administration

...

160 rows selected.

Créer des jointures naturelles

- La clause `NATURAL JOIN` utilise toutes les colonnes des deux tables portant le même nom.
- Elle sélectionne les lignes des deux tables dont les valeurs sont identiques dans toutes les colonnes correspondantes.
- Une erreur est renvoyée lorsque des colonnes portant le même nom présentent des types de données différents.

Extraire des enregistrements à l'aide de jointures naturelles

```
SELECT department_id, department_name,
       location_id, city
FROM   departments
NATURAL JOIN locations ;
```

DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID	CITY
60	IT	1400	Southlake
50	Shipping	1500	South San Francisco
10	Administration	1700	Seattle
90	Executive	1700	Seattle
110	Accounting	1700	Seattle
190	Contracting	1700	Seattle
20	Marketing	1800	Toronto
80	Sales	2500	Oxford

8 rows selected.

Créer des jointures à l'aide de la clause USING

- Si plusieurs colonnes portent le même nom, mais ne possèdent pas le même type de données, la clause `NATURAL JOIN` peut être modifiée à l'aide de la clause `USING` pour indiquer les colonnes à utiliser pour une équijointure.
- La clause `USING` vous permet de n'indiquer qu'une seule colonne lorsque plusieurs colonnes se correspondent.
- N'utilisez pas de nom ou d'alias de table dans les noms des colonnes référencées.
- Les clauses `NATURAL JOIN` et `USING` s'excluent mutuellement.

Extraire des enregistrements à l'aide de la clause USING

```
SELECT e.employee_id, e.last_name, d.location_id
FROM   employees e JOIN departments d
USING (department_id) ;
```

EMPLOYEE_ID	LAST_NAME	LOCATION_ID
200	Whalen	1700
201	Hartstein	1800
202	Fay	1800
124	Mourgos	1500
141	Rajs	1500
142	Davies	1500
143	Matos	1500
144	Vargas	1500
103	Hunold	1400

...

19 rows selected.

Créer des jointures à l'aide de la clause ON

- La condition de la jointure naturelle est une équijointure de toutes les colonnes portant le même nom.
- La clause `ON` permet d'indiquer des conditions arbitraires ou de préciser les colonnes à joindre.
- La condition de jointure est distincte des autres conditions de *recherche*.
- La clause `ON` simplifie la compréhension du code.

Extraire des enregistrements à l'aide de la clause ON

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id);
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
200	Whalen	10	10	1700
201	Hartstein	20	20	1800
202	Fay	20	20	1800
124	Mourgos	50	50	1500
141	Rajs	50	50	1500
142	Davies	50	50	1500
143	Matos	50	50	1500

...

19 rows selected.

Créer des jointures à trois liens à l'aide de la clause ON

```
SELECT employee_id, city, department_name
FROM   employees e
JOIN   departments d
ON     d.department_id = e.department_id
JOIN   locations l
ON     d.location_id = l.location_id;
```

EMPLOYEE_ID	CITY	DEPARTMENT_NAME
103	Southlake	IT
104	Southlake	IT
107	Southlake	IT
124	South San Francisco	Shipping
141	South San Francisco	Shipping
142	South San Francisco	Shipping
143	South San Francisco	Shipping
144	South San Francisco	Shipping

...

19 rows selected.

Jointures INNER et OUTER

- En SQL: 1999, la jointure de deux tables ne renvoyant que les lignes correspondantes est une jointure interne.
- Une jointure entre deux tables renvoyant le résultat de la jointure interne ainsi que les lignes sans correspondance de la table de gauche (ou de droite) est une jointure externe gauche (ou droite).
- Une jointure entre deux tables renvoyant le résultat d'une jointure interne et d'une jointure gauche et droite est une jointure externe complète.

Jointure LEFT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e
LEFT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing

...

De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		

20 rows selected.

Jointure RIGHT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e
RIGHT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
King	90	Executive
Kochhar	90	Executive

...

Whalen	10	Administration
Hartstein	20	Marketing
Fay	20	Marketing
Higgins	110	Accounting
Gietz	110	Accounting
		Contracting

20 rows selected.

Jointure FULL OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e
FULL OUTER JOIN departments d
ON    (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
...		
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		
		Contracting

21 rows selected.

4

Agréger des données à l'aide de fonctions de groupe

Présentation de l'exercice 3, 2ème partie

Dans cet exercice, vous allez :

- joindre des tables à l'aide d'équijointures
- exécuter des jointures externes et des auto-jointures
- ajouter des conditions

Objectifs

- identifier les fonctions de groupe disponibles
- expliquer l'utilisation des fonctions de groupe
- regrouper des données à l'aide de la clause `GROUP BY`
- inclure ou exclure des groupes de lignes à l'aide de la clause `HAVING`

Définition des fonctions de groupe

Les fonctions de groupe agissent sur des groupes de lignes et donnent un résultat par groupe.

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
	7000
10	4400

...

20 rows selected.

Salaire maximum dans la table EMPLOYEES.

MAX(SALARY)
24000

Types de fonction de groupe

- AVG
- COUNT
- MAX
- MIN
- STDDEV
- SUM
- VARIANCE

Syntaxe des fonctions de groupe

```
SELECT      [column,] group_function(column), ...
FROM        table
[WHERE      condition]
[GROUP BY   column]
[ORDER BY   column];
```


Utiliser les fonctions AVG et SUM

Les fonctions AVG et SUM s'utilisent avec des données numériques.

```
SELECT  AVG(salary), MAX(salary),
        MIN(salary), SUM(salary)
FROM    employees
WHERE   job_id LIKE '%REP%';
```

AVG(SALARY)	MAX(SALARY)	MIN(SALARY)	SUM(SALARY)
8150	11000	6000	32600

Utiliser les fonctions MIN et MAX

Les fonctions MIN et MAX s'utilisent avec tous les types de données.

```
SELECT MIN(hire_date), MAX(hire_date)
FROM   employees;
```

MIN(HIRE_	MAX(HIRE_
17-JUN-87	29-JAN-00

Utiliser la fonction COUNT

La fonction COUNT(*) renvoie le nombre de lignes d'une table

```
SELECT COUNT(*)
FROM   employees
WHERE  department_id = 50;
```

COUNT(*)
5

Utiliser la fonction COUNT

- La fonction COUNT(*expr*) renvoie le nombre de lignes contenant des valeurs non NULL dans la colonne *expr*.
- Affichez le nombre de valeurs contenues dans la colonne DEPARTMENT_ID de la table EMPLOYEES, à l'exception des valeurs NULL.

```
SELECT COUNT(commission_pct)
FROM   employees
WHERE  department_id = 80;
```

COUNT(COMMISSION_PCT)
3

Utiliser le mot-clé DISTINCT

- La fonction COUNT(DISTINCT expr) renvoie le nombre de valeurs non NULL distinctes de la colonne *expr*.
- Affichez le nombre de services distincts contenus dans la table EMPLOYEES.

```
SELECT COUNT(DISTINCT department_id)
FROM   employees;
```

COUNT(DISTINCTDEPARTMENT_ID)
7

Fonctions de groupe et valeurs NULL

Les fonctions de groupe ignorent les valeurs NULL des colonnes.

```
SELECT AVG(commission_pct)
FROM   employees;
```

AVG(COMMISSION_PCT)
.2125

Utiliser la fonction NVL avec les fonctions de groupe

La fonction NVL contraint les fonctions de groupe à intégrer des valeurs NULL.

```
SELECT AVG(NVL(commission_pct, 0))
FROM   employees;
```

AVG(NVL(COMMISSION_PCT,0))
.0425

Créer des groupes de données

EMPLOYEES

DEPARTMENT_ID	SALARY
10	4400
20	13000
20	6000
50	5800
50	3500
50	3100
50	2500
50	2600
60	9000
60	6000
60	4200
80	10500
80	8600
80	11000
90	24000
90	17000

...

20 rows selected.

4400
9500
3500
6400
10033

Salaire moyen par service dans la table EMPLOYEES.

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

Créer des groupes de données : syntaxe de la clause GROUP BY

```
SELECT      column, group_function(column)
FROM        table
[WHERE      condition]
[GROUP BY   group_by_expression]
[ORDER BY   column];
```

La clause GROUP BY permet d'organiser les lignes d'une table en groupes restreints.

Utiliser la clause GROUP BY

La clause GROUP BY doit inclure toutes les colonnes de la liste SELECT qui ne figurent pas dans des fonctions de groupe.

```
SELECT    department_id, AVG(salary)
FROM      employees
GROUP BY department_id ;
```

DEPARTMENT_ID	AVG(SALARY)
10	4400
20	9500
50	3500
60	6400
80	10033.3333
90	19333.3333
110	10150
	7000

8 rows selected.

Utiliser la clause GROUP BY

La colonne GROUP BY ne doit pas nécessairement figurer dans la liste SELECT.

```
SELECT    AVG(salary)
FROM      employees
GROUP BY department_id ;
```

AVG(SALARY)
4400
9500
3500
6400
10033.3333
19333.3333
10150
7000

Créer des sous-groupes

EMPLOYEES

DEPARTMENT_ID	JOB_ID	SALARY
90	AD_PRES	24000
90	AD_VP	17000
90	AD_VP	17000
60	IT_PROG	9000
60	IT_PROG	6000
60	IT_PROG	4200
50	ST_MAN	5800
50	ST_CLERK	3500
50	ST_CLERK	3100
50	ST_CLERK	2600
50	ST_CLERK	2500
80	SA_MAN	10500
80	SA_REP	11000
80	SA_REP	8600
...		
20	MK_REP	6000
110	AC_MGR	12000
110	AC_ACCOUNT	8300

20 rows selected.

Dans la table EMPLOYEES, calcul du total des salaires pour chaque poste, au sein de chaque service.

DEPARTMENT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD_PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected.

Utiliser la clause GROUP BY sur plusieurs colonnes

```
SELECT    department_id dept_id, job_id, SUM(salary)
FROM      employees
GROUP BY department_id, job_id ;
```

DEPT_ID	JOB_ID	SUM(SALARY)
10	AD_ASST	4400
20	MK_MAN	13000
20	MK_REP	6000
50	ST_CLERK	11700
50	ST_MAN	5800
60	IT_PROG	19200
80	SA_MAN	10500
80	SA_REP	19600
90	AD_PRES	24000
90	AD_VP	34000
110	AC_ACCOUNT	8300
110	AC_MGR	12000
	SA_REP	7000

13 rows selected.

Erreurs d'utilisation des fonctions de groupe dans une interrogation

Toute colonne ou expression de la liste SELECT autre qu'une fonction d'agrégation doit être incluse dans la clause GROUP BY.

```
SELECT department_id, COUNT(last_name)
FROM   employees;
```

```
SELECT department_id, COUNT(last_name)
       *
ERROR at line 1:
ORA-00937: not a single-group group function
```

Colonne manquante dans la clause GROUP BY

Erreurs d'utilisation des fonctions de groupe dans une interrogation

- Vous ne pouvez pas utiliser la clause WHERE pour limiter les groupes.
- Utilisez la clause HAVING.
- Vous ne pouvez pas utiliser de fonctions de groupe dans la clause WHERE.

```
SELECT   department_id, AVG(salary)
FROM     employees
WHERE    AVG(salary) > 8000
GROUP BY department_id;
```

```
WHERE  AVG(salary) > 8000
       *
ERROR at line 3:
ORA-00934: group function is not allowed here
```

N'utilisez pas la clause WHERE pour limiter les groupes

Exclure des groupes de résultats

EMPLOYEES

DEPARTMENT_ID	SALARY
90	24000
90	17000
90	17000
60	9000
60	6000
60	4200
50	5800
50	3500
50	3100
50	2600
50	2500
80	10500
80	11000
80	8600
...	
20	6000
110	12000
110	8300

20 rows selected.

Salaire maximum par service, à condition qu'il soit supérieur à 10 000 $

DEPARTMENT_ID	MAX(SALARY)
20	13000
80	11000
90	24000
110	12000

Exclure des groupes de résultats : clause HAVING

Utilisez la clause HAVING pour restreindre les groupes.

1. Les lignes sont regroupées.
2. La fonction de groupe est appliquée.
3. Les groupes qui correspondent à la clause HAVING s'affichent.

```
SELECT        column, group_function
FROM          table
[WHERE        condition]
[GROUP BY     group_by_expression]
[HAVING       group_condition]
[ORDER BY     column];
```

Utiliser la clause HAVING

```
SELECT    department_id, MAX(salary)
FROM      employees
GROUP BY department_id
HAVING    MAX(salary)>10000 ;
```

DEPARTMENT_ID	MAX(SALARY)
20	13000
60	11000
90	24000
110	12000

Utiliser la clause HAVING

```
SELECT    job_id, SUM(salary) PAYROLL
FROM      employees
WHERE     job_id NOT LIKE '%REP%'
GROUP BY job_id
HAVING    SUM(salary) > 13000
ORDER BY SUM(salary);
```

JOB_ID	PAYROLL
IT_PROG	19200
AD_PRES	24000
AD_VP	34000

Imbriquer des fonctions de groupe

Affichez le salaire moyen maximum.

```
SELECT    MAX(AVG(salary))
FROM      employees
GROUP BY department_id;
```

MAX(AVG(SALARY))
19333.3333

Synthèse

- utiliser les fonctions de groupe COUNT, MAX, MIN, AVG
- écrire des instructions contenant la clause GROUP BY
- écrire des intructions contenant la clause HAVING

```
SELECT        column, group_function(column)
FROM          table
[WHERE        condition]
[GROUP BY     group_by_expression]
[HAVING       group_condition]
[ORDER BY     column];
```

Présentation de l'exercice 5

Dans cet exercice, vous allez :

- écrire des instructions contenant des fonctions de groupe
- grouper des lignes pour obtenir plusieurs résultats
- exclure des groupes en utilisant la clause `HAVING`

Objectifs

A la fin de ce chapitre, vous pourrez :

- décrire chaque instruction LMD
- insérer des lignes dans une table
- modifier des lignes dans une table
- supprimer des lignes d'une table
- fusionner des lignes dans une table
- contrôler les transactions

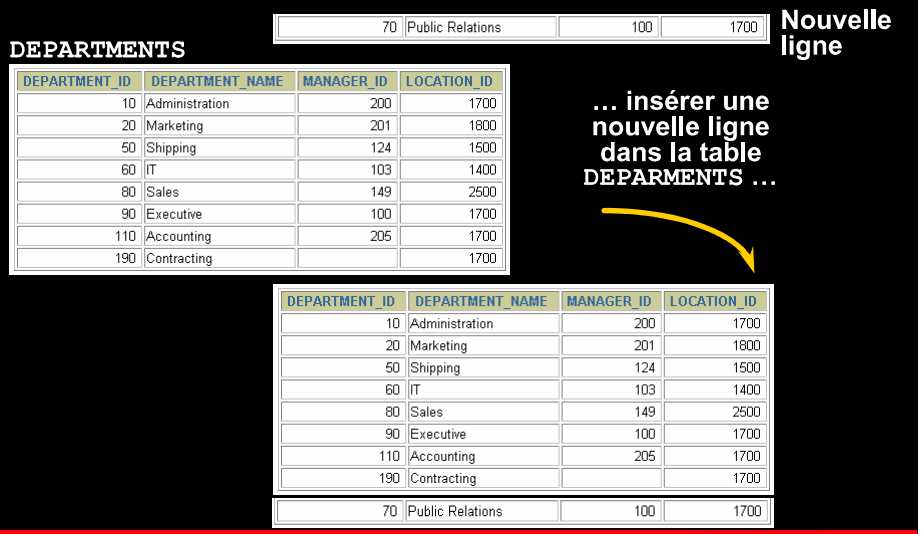
5

Manipuler des données

Langage de manipulation de données

- Une instruction LMD est exécutée lorsque vous :
  - ajoutez des lignes à une table,
  - modifiez des lignes existantes dans une table,
  - supprimez des lignes d'une table.
- Une *transaction* est un ensemble d'instructions LMD formant une unité de travail logique.

Ajouter une nouvelle ligne dans une table



Syntaxe de l'instruction INSERT

- L'instruction INSERT permet d'ajouter de nouvelles lignes dans une table.

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

- Cette syntaxe n'insère qu'une seule ligne à la fois.

Insérer de nouvelles lignes

- Insérez une nouvelle ligne en précisant une valeur pour chaque colonne.
- Indiquez les valeurs dans l'ordre par défaut des colonnes dans la table.
- Indiquez éventuellement les colonnes dans la clause INSERT.

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

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

Insérer des lignes contenant des valeurs NULL

- Méthode implicite : n'indiquez pas la colonne dans la liste.

```
INSERT INTO   departments (department_id,
                           department_name )
VALUES        (30, 'Purchasing');
1 row created.
```

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

```
INSERT INTO   departments
VALUES        (100, 'Finance', NULL, NULL);
1 row created.
```

Insérer des valeurs spéciales

La fonction SYSDATE enregistre la date et l'heure en cours.

```
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, 100);
1 row created.
```


Insérer des dates dans un format spécifique

- Ajoutez un nouvel employé.

```
INSERT INTO employees
VALUES      (114,
             'Den', 'Raphealy',
             'DRAPHEAL', '515.127.4561',
             TO_DATE('FEB 3, 1999', 'MON DD, YYYY'),
             'AC_ACCOUNT', 11000, NULL, 100, 30);
1 row created.
```

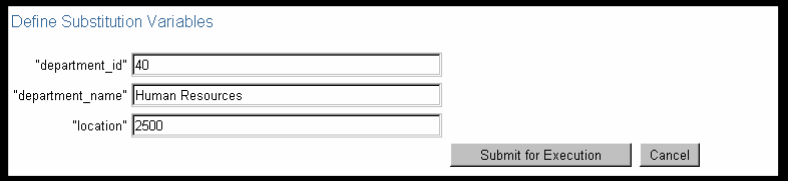
- Vérifiez l'ajout.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_P
114	Den	Raphealy	DRAPHEAL	515.127.4561	03-FEB-99	AC_ACCOUNT	11000	

Créer un script

- Utilisez le caractère de substitution & dans une instruction SQL de saisie de valeurs.
- & est une marque de réservation pour les variables.

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




```
1 row created.
```


Copier des lignes d'une autre table

- Ecrivez votre instruction INSERT en précisant 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 created.
```

- N'utilisez pas la clause VALUES.
- Le nombre de colonnes de la clause INSERT doit correspondre à celui de la sous-interrogation.

Modifier les données d'une table

EMPLOYEES

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSION_P
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	60	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	60	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	60	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

Mettez à jour les lignes de la table **EMPLOYEES**.

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID	COMMISSIO
100	Steven	King	SKING	17-JUN-87	AD_PRES	24000	90	
101	Neena	Kochhar	NKOCHHAR	21-SEP-89	AD_VP	17000	90	
102	Lex	De Haan	LDEHAAN	13-JAN-93	AD_VP	17000	90	
103	Alexander	Hunold	AHUNOLD	03-JAN-90	IT_PROG	9000	30	
104	Bruce	Ernst	BERNST	21-MAY-91	IT_PROG	6000	30	
107	Diana	Lorentz	DLORENTZ	07-FEB-99	IT_PROG	4200	30	
124	Kevin	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50	

Syntaxe de l'instruction UPDATE

- Utilisez l'instruction **UPDATE** pour modifier des lignes existantes.

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

- Si nécessaire, vous pouvez modifier plusieurs lignes à la fois.

Modifier des lignes d'une table

- La clause **WHERE** permet de modifier une ou plusieurs lignes spécifiques.

```
UPDATE employees
SET    department_id = 70
WHERE  employee_id = 113;
1 row updated.
```

- En cas d'absence de la clause **WHERE**, toutes les lignes sont modifiées.

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


Modifier deux colonnes à l'aide d'une sous-interrogation

Modifiez le poste et le salaire de l'employé 114 pour 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    =  114;
1 row updated.
```

Modifier des lignes en fonction d'une autre table

Utilisez des sous-interrogations dans l'instruction `UPDATE` pour modifier des lignes d'une table à l'aide de 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 row updated.
```


Erreur de contrainte d'intégrité lors de la modification de lignes

```
UPDATE employees
SET    department_id = 55
WHERE  department_id = 110;
```

```
UPDATE employees
       *
ERROR at line 1:
ORA-02291: integrity constraint (HR.EMP_DEPT_FK)
violated - parent key not found
```

Le numéro de service 55 n'existe pas.

Supprimer une ligne d'une table

`DEPARTMENTS`

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
100	Finance		
50	Shipping	124	1500
60	IT	103	1400

Supprimez une ligne de la table `DEPARTMENTS`.

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing		
50	Shipping	124	1500
60	IT	103	1400

Instruction `DELETE`

Vous pouvez supprimer des lignes d'une table au moyen de l'instruction `DELETE`.

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

Supprimer des lignes d'une table

- La clause `WHERE` permet de supprimer des lignes spécifiques.

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

- En cas d'absence de la clause `WHERE`, toutes les lignes sont supprimées.

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


Supprimer des lignes associées à des valeurs d'une autre table

Utilisez des sous-interrogations dans l'instruction `DELETE` pour supprimer des lignes dont certaines valeurs correspondent à celles d'une autre table.

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


Erreur de contrainte d'intégrité lors de la suppression de lignes

```
DELETE FROM departments
WHERE       department_id = 60;
```

```
DELETE FROM departments
            *
ERROR at line 1:
ORA-02292: integrity constraint (HR.EMP_DEPT_FK)
violated - child record found
```

Vous ne pouvez pas supprimer une ligne qui contient une clé primaire utilisée comme clé étrangère dans une autre table.

Utiliser une sous-interrogation dans une instruction `INSERT`

```
INSERT INTO
        (SELECT employee_id, last_name,
                email, hire_date, job_id, salary,
                department_id
         FROM   employees
         WHERE  department_id = 50)
VALUES (99999, 'Taylor', 'DTAYLOR',
        TO_DATE('07-JUN-99', 'DD-MON-RR'),
        'ST_CLERK', 5000, 50);

1 row created.
```

Utiliser une sous-interrogation dans une instruction INSERT

```
SELECT employee_id, last_name, email, hire_date,
       job_id, salary, department_id
FROM   employees
WHERE  department_id = 50;
```

EMPLOYEE_ID	LAST_NAME	EMAIL	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
124	Mourgos	KMOURGOS	16-NOV-99	ST_MAN	5800	50
141	Rajs	TRAJS	17-OCT-95	ST_CLERK	3500	50
142	Davies	CDAVIES	29-JAN-97	ST_CLERK	3100	50
143	Matos	RMATOS	15-MAR-98	ST_CLERK	2600	50
144	Vargas	PVARGAS	09-JUL-98	ST_CLERK	2500	50
99999	Taylor	DTAYLOR	07-JUN-99	ST_CLERK	5000	50

6 rows selected.

Utiliser le mot-clé WITH CHECK OPTION avec les instructions LMD

- Une sous-interrogation permet d'identifier la table et les colonnes des instructions LMD.
- Le mot-clé WITH CHECK OPTION vous empêche de modifier les lignes qui ne sont pas présentes dans la sous-interrogation.

```
INSERT INTO  (SELECT employee_id, last_name, email,
                hire_date, job_id, salary
              FROM   employees
              WHERE  department_id = 50 WITH CHECK OPTION)
VALUES (99998, 'Smith', 'JSMITH',
        TO_DATE('07-JUN-99', 'DD-MON-RR'),
        'ST_CLERK', 5000);
INSERT INTO
           *
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation
```


Valeur par défaut explicite : présentation

- La fonction de définition de valeur par défaut explicite vous permet d'utiliser le mot-clé DEFAULT en tant que valeur de colonne lorsque vous avez besoin d'une valeur de colonne par défaut.
- L'intégration de cette fonction permet la conformité à la norme SQL: 1999.
- L'utilisateur peut ainsi contrôler quand et où la valeur par défaut doit être appliquée aux données.
- Les valeurs par défaut explicites peuvent être utilisées dans les instructions INSERT et UPDATE.

Utiliser des valeurs explicites par défaut

- DEFAULT avec INSERT :

```
INSERT INTO departments
  (department_id, department_name, manager_id)
VALUES (300, 'Engineering', DEFAULT);
```

- DEFAULT avec UPDATE :

```
UPDATE departments
SET manager_id = DEFAULT WHERE department_id = 10;
```

Instruction MERGE

- Permet de mettre à jour ou d'insérer des données dans une table, de façon conditionnelle.
- Exécute une instruction UPDATE si la ligne existe et une instruction INSERT s'il s'agit d'une nouvelle ligne :
  - Evite des mises à jour distinctes
  - Améliore les performances et facilite l'utilisation
  - S'avère particulièrement utile dans les applications de data warehouse

Syntaxe de l'instruction MERGE

L'instruction MERGE vous permet de mettre à jour ou d'insérer des lignes dans une table de façon conditionnelle.

```
MERGE INTO table_name table_alias
  USING (table|view|sub_query) alias
  ON (join condition)
  WHEN MATCHED THEN
    UPDATE SET
    col1 = col_val1,
    col2 = col2_val
  WHEN NOT MATCHED THEN
    INSERT (column_list)
    VALUES (column_values);
```


Fusionner des lignes

Insérez ou mettez à jour des lignes dans la table COPY_EMP pour qu'elle corresponde à la table EMPLOYEES.

```
MERGE INTO copy_emp  c
  USING employees e
  ON (c.employee_id = e.employee_id)
WHEN MATCHED THEN
  UPDATE SET
     c.first_name      = e.first_name,
     c.last_name       = e.last_name,
     ...
     c.department_id   = e.department_id
WHEN NOT MATCHED THEN
 INSERT VALUES(e.employee_id, e.first_name, e.last_name,
          e.email, e.phone_number, e.hire_date, e.job_id,
          e.salary, e.commission_pct, e.manager_id,
          e.department_id);
```


Fusionner des lignes

```
SELECT *
FROM COPY_EMP;

no rows selected
```

```
MERGE INTO copy_emp c
  USING employees e
  ON (c.employee_id = e.employee_id)
WHEN MATCHED THEN
  UPDATE SET
     ...
WHEN NOT MATCHED THEN
 INSERT VALUES...;
```

```
SELECT *
FROM COPY_EMP;

20 rows selected.
```

Transactions de la base de données

Une transaction de base de données est constituée de l'un des éléments suivants :

- des instructions LMD effectuant une modification cohérente des données,
- une instruction LDD,
- une instruction LCD.

Transactions de la base de données

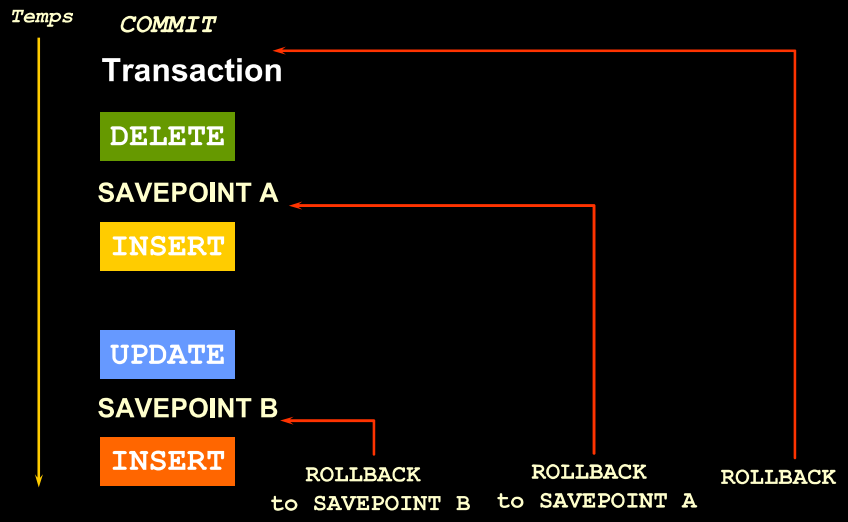
- Commence à l'exécution de la première instruction LMD SQL.
- Se termine par l'un des événements suivants :
  - Une instruction COMMIT ou ROLLBACK est lancée.
  - Une instruction LDD ou LCD (validation automatique) est exécutée.
  - L'utilisateur quitte *i*SQL*Plus.
  - Le système tombe en panne.

Avantages des instructions COMMIT et ROLLBACK

Les instructions COMMIT et ROLLBACK vous permettent :

- de garantir la cohérence des données,
- d'afficher le résultat des modifications de données avant qu'elles ne soient définitives,
- de regrouper de manière logique des opérations associées.

Contrôler les transactions



Annuler des modifications jusqu'à une étiquette

- Créez une étiquette dans la transaction courante à l'aide de l'instruction SAVEPOINT.
- Annulez la transaction jusqu'à cette étiquette en utilisant l'instruction ROLLBACK TO SAVEPOINT.

```
UPDATE...
SAVEPOINT update_done;
Savepoint created.
INSERT...
ROLLBACK TO update_done;
Rollback complete.
```


Traitement implicite des transactions

- Une validation automatique a lieu dans les situations suivantes :
  - exécution d'une instruction LDD,
  - exécution d'une instruction LCD,
  - sortie normale d'*i*SQL*Plus, sans instruction COMMIT ou ROLLBACK explicite.
- Il se produit une annulation automatique en cas de sortie anormale d'*i*SQL*Plus ou d'une panne du système.

Etat des données avant COMMIT ou ROLLBACK

- Il est possible de restaurer l'état précédent des données.
- L'utilisateur en cours peut afficher le résultat des opérations LMD au moyen de l'instruction SELECT.
- Les résultats des instructions LMD exécutées par l'utilisateur courant *ne peuvent pas* être affichés par d'autres utilisateurs.
- Les lignes concernées sont *verrouillées*. Aucun autre utilisateur ne peut les modifier.

Etat des données après COMMIT

- Les modifications des données dans la base sont définitives.
- L'état précédent des données est irrémédiablement perdu.
- Tous les utilisateurs peuvent voir le résultat des modifications.
- Les lignes verrouillées sont libérées et peuvent de nouveau être manipulées par d'autres utilisateurs.
- Tous les savepoints sont effacés.

Valider des données

- Effectuez les modifications.

```
DELETE FROM employees
WHERE  employee_id = 99999;
1 row deleted.

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

- Validez les modifications.

```
COMMIT;
Commit complete.
```


Etat des données après ROLLBACK

L'instruction ROLLBACK permet de rejeter toutes les modifications de données en cours :

- Les modifications sont annulées.
- Les données retrouvent leur état précédent.
- Les lignes verrouillées sont libérées.

```
DELETE FROM copy_emp;
22 rows deleted.
ROLLBACK;
Rollback complete.
```

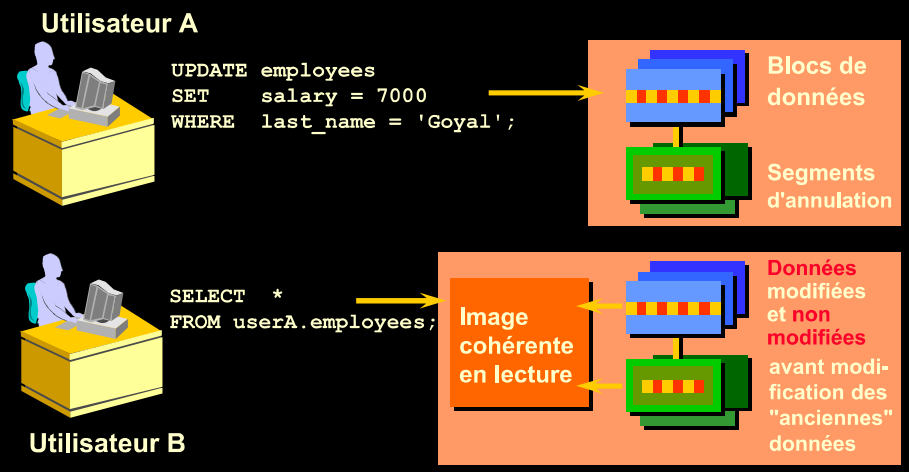

Annulation au niveau instruction

- Si une instruction LMD échoue pendant l'exécution, seule cette instruction est annulée.
- Le serveur Oracle met en oeuvre un savepoint implicite.
- Toutes les autres modifications sont conservées.
- L'utilisateur doit terminer explicitement les transactions en exécutant une instruction COMMIT ou ROLLBACK.

Cohérence en lecture

- La cohérence en lecture garantit à tout moment une vue homogène des données.
- Les modifications apportées par un utilisateur n'entrent pas en conflit avec celles d'un autre utilisateur.
- La cohérence en lecture garantit sur les mêmes données que :
  - la lecture ignore les écritures en cours,
  - l'écriture ne perturbe pas la lecture.

Implémenter la cohérence en lecture



Verrouillage

Les verrous d'une base de données Oracle :

- évitent les risques de destruction des données en cas de transactions simultanées,
- n'exigent aucune intervention de l'utilisateur,
- s'appliquent au niveau de restriction le plus bas,
- sont actifs durant toute la transaction,
- sont de deux types : explicite et implicite.

Verrouillage implicite

- Deux modes de verrouillage :
  - Verrou de type Exclusive : Verrouille l'accès à tous les autres utilisateurs
  - Verrou de type Share : Permet aux autres utilisateurs d'accéder également aux données
- Haut niveau de simultanéité d'accès aux données
  - LMD : Partage des tables, row exclusive
  - Interrogations : Aucun verrou requis
  - LDD : Protège les définitions d'objet
- Verrouillage maintenu jusqu'à validation ou annulation

Synthèse

Ce chapitre vous a permis d'apprendre à utiliser des instructions LMD et à contrôler les transactions.

Instruction	Description
INSERT	Ajoute une nouvelle ligne dans une table
UPDATE	Modifie des lignes dans une table
DELETE	Supprime des lignes d'une table
MERGE	Insère ou met à jour des données dans une table de façon conditionnelle
COMMIT	Valide toutes les modifications de données en cours jusqu'à l'étiquette du savepoint
SAVEPOINT	Permet une annulation jusqu'à un savepoint
ROLLBACK	Annule toutes les modifications de données en instance

Présentation de l'exercice 8

Dans cet exercice, vous allez :

- insérer des lignes dans une table
- mettre à jour et supprimer des lignes dans une table
- contrôler des transactions

Jointure FULL OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e
FULL OUTER JOIN departments d
ON    (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
...		
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		
		Contracting

21 rows selected.

Autres conditions

```
SELECT e.employee_id, e.last_name, e.department_id,
       d.department_id, d.location_id
FROM   employees e JOIN departments d
ON     (e.department_id = d.department_id)
AND    e.manager_id = 149 ;
```

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_ID	LOCATION_ID
174	Abel	80	80	2500
176	Taylor	80	80	2500

Présentation de l'exercice 4, 2ème partie

Dans cet exercice, vous allez :

- joindre des tables à l'aide d'équijointures
- exécuter des jointures externes et des auto-jointures
- ajouter des conditions


ORACLE