# 4

# Afficher des données issues de plusieurs tables

# Objectifs

A la fin de ce chapitre, vous pourrez :

- 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

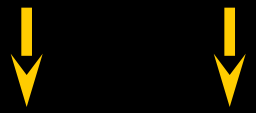
# Afficher des données issues de plusieurs tables

**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 propriétaires Oracle (version 8*i* et précédentes) :**

- Equijointure
- Non-équijointure
- Jointure externe
- Auto-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**

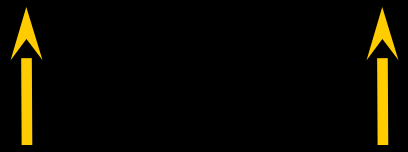
| EMPLOYEE_ID | DEPARTMENT_ID |
|---|---|
| 200 | 10 |
| 201 | 20 |
| 202 | 20 |
| 124 | 50 |
| 141 | 50 |
| 142 | 50 |
| 143 | 50 |
| 144 | 50 |
| 103 | 60 |
| 104 | 60 |
| 107 | 60 |
| 149 | 80 |
| 174 | 80 |
| 176 | 80 |

...

**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 |

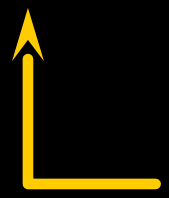
8 rows selected.

**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 4, 1ère partie

**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.

# 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 |

# Synthèse

Ce chapitre vous a permis d'apprendre à utiliser des jointures afin d'afficher des données provenant de plusieurs tables en respectant :

- la syntaxe propriétaire Oracle pour les versions 8*i* et antérieures,
- *la syntaxe conforme à la norme SQL: 1999 pour la version 9i.*

# 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