Afficher des données de plusieurs tables

Objectifs

A la fin de ce TP, vous pourrez :

- **écrire des instructions `SELECT` afin d'accéder aux données de plusieurs tables à l'aide d'équijointures et de non-équijointures**
- **joindre une table à elle-même à l'aide d'une auto-jointure**
- **afficher des données qui ne satisfont généralement pas à une condition de jointure, à l'aide de jointures externes**
- **générer un produit cartésien de toutes les lignes de plusieurs tables**

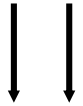
Obtenir des données de plusieurs tables

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

Types de jointure

Les jointures conformes à la norme SQL:1999 sont :

- **Jointures naturelles** `Natural Join`
- **Clause** `USING` `Join … Using`
- **Clause** `ON` `Join … On`
- **Auto-jointures**
- **Non équijointures**
- **Jointures externes (gauche ; droite ; complète)**
- **Jointures croisées**

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

Utilisez une jointure pour interroger des données provenant de plusieurs tables :

```
SELECT   table1.column, table2.column
FROM     table1
[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)]|
[CROSS JOIN table2];
```

Créer des jointures naturelles

- La clause `NATURAL JOIN` est basée sur 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 qui correspondent.
- Si les colonnes portant le même nom présentent des types de données différents, une erreur est renvoyée.

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 avec la clause USING

- Si plusieurs colonnes portent le même nom, et on veut privilégier une colonne particulière pour la jointure, la clause NATURAL JOIN est replacée par la clause USING, laquelle permet de désigner la colonne qui doit être utilisée pour la jointure.
- Utilisez donc la clause USING pour n'indiquer qu'une seule colonne lorsque plusieurs colonnes correspondent.
- N'utilisez pas de nom ou alias de table dans les colonnes référencées.
- Les clauses NATURAL JOIN et USING sont mutuellement exclusives.

Joindre des noms de colonne

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 avec la clause `USING`

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

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

...

19 rows selected.

Différencier les noms de colonne

- Utilisez des préfixes qui précisent le nom de la table pour différencier les noms de colonne présents dans plusieurs tables.
- L'utilisation de préfixes désignant la table améliore les performances.
- Utilisez des alias de colonne pour distinguer les colonnes qui présentent des noms identiques, mais qui résident dans des tables différentes.
- N'utilisez pas d'alias sur les colonnes identifiées dans la clause `USING` et indiquées ailleurs dans l'instruction SQL.

Différencier les noms de colonne

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

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

...

19 rows selected.

Utiliser des alias de table

- Utilisez des alias de table pour simplifier les interrogations.
- Utilisez des alias de table pour améliorer les performances.

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

Créer des jointures avec la clause ON

- La condition de la jointure naturelle est une équijointure de toutes les colonnes portant le même nom.
- Utilisez la clause ON pour indiquer des conditions arbitraires ou pour désigner les colonnes à joindre.
- La condition de jointure est distincte des autres conditions de recherche.
- La clause ON facilite la compréhension du code.

Extraire des enregistrements avec 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.

Auto-jointures avec la clause ON

EMPLOYEES (WORKER)

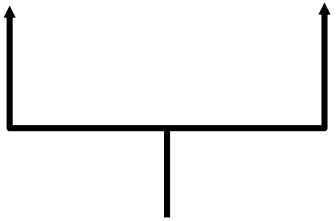
EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

...

EMPLOYEES (MANAGER)

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

...



MANAGER_ID dans la table WORKER est égal à EMPLOYEE_ID dans la table MANAGER.

Auto-jointures avec la clause ON

```
SELECT e.last_name emp, m.last_name mgr
FROM   employees e JOIN employees m
ON     (e.manager_id = m.employee_id);
```

EMP	MGR
Hartstein	King
Zlotkey	King
Mourgos	King
De Haan	King
Kochhar	King

...

19 rows selected.

Appliquer des conditions supplémentaires à une jointure

```
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

Créer des jointures à trois liens avec 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.

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

Le salaire de la table EMPLOYEES doit être compris entre le salaire le plus faible et le salaire le plus élevé 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 JOIN job_grades j
ON     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 département 190 ne compte aucun employé.

Comparaison des jointures INNER et OUTER

- Dans la syntaxe SQL : 1999, la jointure de deux tables qui renvoie uniquement les lignes qui correspondent est une jointure interne.
- Une jointure entre deux tables qui renvoie les résultats de la jointure interne, ainsi que les lignes de la table de gauche (ou droite) qui ne correspondent pas, est une jointure externe gauche (ou droite).
- Une jointure entre deux tables qui renvoie les résultats d'une jointure interne, ainsi que les résultats d'une jointure gauche et droite, est une jointure externe complète.

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.

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
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
Davies	50	Shipping
...		
Kochhar	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
	190	Contracting

20 rows selected.

FULL OUTER JOIN

```
SELECT e.last_name, d.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
Hartstein	20	Marketing
...		
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		
	190	Contracting

21 rows selected.

Produits cartésiens

- **Un produit cartésien est généré dans les cas suivants :**
  - **Une condition de jointure est omise**
  - **Une condition de jointure n'est pas valide**
  - **Toutes les lignes de la première table sont jointes à toutes les lignes de la seconde**
- **Pour éviter un produit cartésien, incluez toujours une condition de jointure valide.**

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 :
20 x 8 = 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.

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.

Synthèse

Dans ce TP, vous avez appris à utiliser des jointures pour afficher des données de plusieurs tables avec des :

- **jointures naturelles**
- **auto-jointures**
- **non-équijointures**
- **jointures externes (gauche ; droite ; complète)**
- **jointures croisées**