



Restreindre et trier les données

Objectifs

A la fin de ce TP, vous pourrez :

- **limiter les lignes extraites par une interrogation**
- **trier les lignes extraites par une interrogation**

Limiter les 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
département 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 les lignes sélectionnées

- Restreindre les lignes renvoyées à l'aide de la clause WHERE :

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

- La clause WHERE suit 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 sont incluses entre apostrophes.
- Les valeurs de type caractère distinguent les majuscules des minuscules et les valeurs de date sont sensibles au format.
- Le format de date par défaut est DD-MON-YY.

```
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
<>	Non égal à
BETWEEN ...AND...	Entre deux valeurs (incluses)
IN (set)	Correspond à une valeur quelconque d'une liste
LIKE	Correspond à un modèle de caractère
IS NULL	Est une valeur NULL

Utiliser des conditions de comparaison

```
SELECT last_name, salary
FROM employees
WHERE salary <= 3000 ;
```

LAST_NAME	SALARY
Matos	2600
Vargas	2500

Utiliser la condition BETWEEN

Utilisez la condition BETWEEN pour afficher les 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 afin de tester les valeurs d'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 effectuer des recherches de chaînes de caractères valides via l'utilisation de caractères génériques.
- Les conditions de recherche peuvent contenir soit des caractères littéraux, soit des nombres :
 - % indique un nombre quelconque de caractères (zéro ou plus).
 - _ indique un caractère unique.

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

Utiliser la condition LIKE

- Vous pouvez combiner des caractères de mise en correspondance de modèle :

```
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 _ proprement dits.

```
SELECT employee_id, last_name, job_id
FROM employees
WHERE job_id LIKE 'SA\_%' ESCAPE '\\';
```

Utiliser la condition NULL

Testez la présence de 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 <i>les deux</i> conditions sont vraies.
OR	Renvoie TRUE si <i>l'une des deux</i> conditions est vraie.
NOT	Renvoie TRUE si la condition qui suit est fausse.

Utiliser l'opérateur AND

L'opérateur AND nécessite 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 nécessite que l'une des deux 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

Opérateur	Signification
1	Opérateurs arithmétiques
2	Opérateur de concaténation
3	Conditions de comparaison
4	IS [NOT] NULL, LIKE, [NOT] IN
5	[NOT] BETWEEN
6	Non égal à
7	Condition logique NOT
8	Condition logique AND
9	Condition logique OR

Vous pouvez utiliser des parenthèses pour remplacer des 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;
```

1

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

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

2

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000

Utiliser la clause ORDER BY

- Trier les lignes extraites à l'aide de la clause ORDER BY :
 - ASC : ordre croissant (par défaut)
 - DESC : ordre décroissant
- La clause ORDER BY vient en dernier dans 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

- Trier par ordre décroissant :

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

1

- Trier par alias de colonne :

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

2

- Trier selon plusieurs colonnes :

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

3

Variables de substitution



Variables de substitution

- **Utilisez les variables de substitution pour :**
 - Stocker temporairement des valeurs, via l'esperluette d'interprétation (&) et la double esperluette d'interprétation (&&)
- **Utilisez des variables de substitution en complément des éléments suivants :**
 - Conditions WHERE
 - Clauses ORDER BY
 - Expressions de colonne
 - Noms des tables
 - Instructions SELECT entières

Utiliser la variable de substitution &

Utilisez une variable précédée d'une esperluette (&) afin d'inviter l'utilisateur à saisir une valeur :

```
SELECT employee_id, last_name, salary, department_id
FROM   employees
WHERE  employee_id = &employee_num ;
```

Connected as **ORA1@T6**

 **Input Required**

Enter value for employee_num:

Utiliser la variable de substitution &

 **Input Required**

Enter value for employee_num:

 **1**

 **2**

old 3: WHERE employee_id = &employee_num

new 3: WHERE employee_id = 101

EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
101	Kochhar	17000	90

Valeurs de type caractère et date avec des variables de substitution

Utilisez des apostrophes pour les valeurs de type date et caractère :

```
SELECT last_name, department_id, salary*12
FROM employees
WHERE job_id = '&job_title' ;
```



Input Required

Cancel

Continue

Enter value for job_title: IT_PROG

LAST_NAME	DEPARTMENT_ID	SALARY*12
Hunold	60	108000
Ernst	60	72000
Lorentz	60	50400

Définir des noms de colonne, des expressions et du texte

```
SELECT employee_id, last_name, job_id, &column_name  
FROM employees  
WHERE &condition  
ORDER BY &order_column ;
```

Input Required

Cancel

Continue

Enter value for column_name:

Cancel

Continue

Enter value for condition:

Cancel


Continue

Enter value for order_column:

Utiliser la variable de substitution &&

Utilisez la double esperluette (&&) si vous souhaitez réutiliser la valeur de la variable sans inviter chaque fois l'utilisateur à la saisir :

```
SELECT  employee_id, last_name, job_id, &&column_name
FROM    employees
ORDER BY &column_name ;
```

 **Input Required**

Cancel Continue

Enter value for column_name:

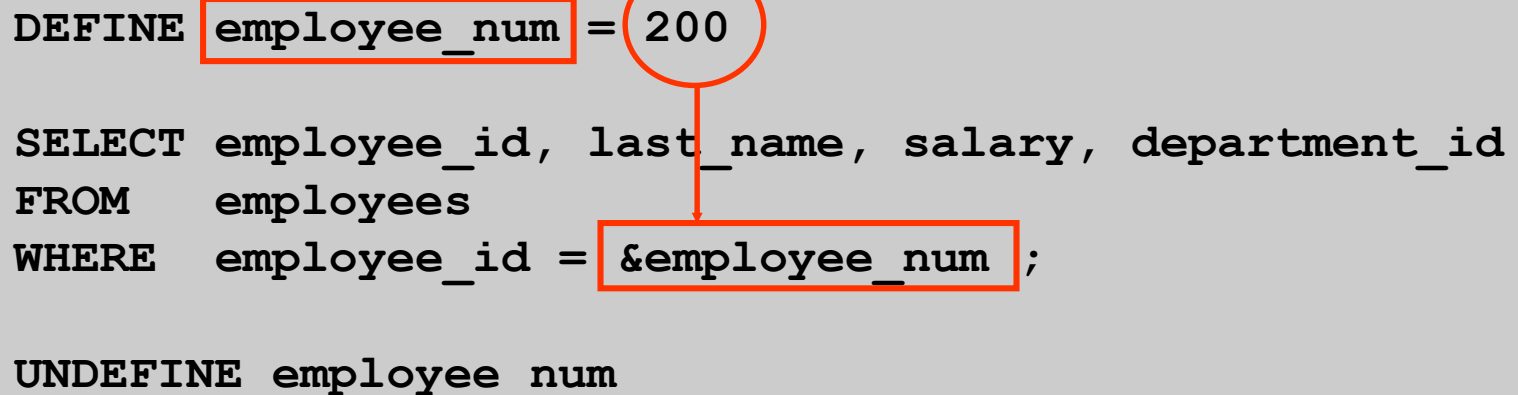
EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
200	Whalen	AD_ASST	10
201	Hartstein	MK_MAN	20

...
20 rows selected.

Utiliser la commande SQL*Plus DEFINE

- Utilisez la commande SQL*Plus DEFINE pour créer une valeur et pour l'affecter à une variable.
- Utilisez la commande SQL*Plus UNDEFINE pour supprimer une variable.

```
DEFINE employee_num = 200  
  
SELECT employee_id, last_name, salary, department_id  
FROM employees  
WHERE employee_id = &employee_num;  
  
UNDEFINE employee_num
```



Utiliser la commande VERIFY

Utilisez la commande VERIFY pour basculer l'affichage de la variable de substitution, à la fois avant et après le remplacement par SQL*Plus des variables de substitution par des valeurs :

```
SET VERIFY ON
```

```
SELECT employee_id, last_name, salary, department_id  
FROM   employees  
WHERE  employee_id = &employee_num;
```

"employee_num" 200

```
old      3: WHERE  employee_id = &employee_num  
new      3: WHERE  employee id = 200
```

Synthèse

Ce TP vous a permis d'apprendre à :

- utiliser la clause **WHERE** pour restreindre les lignes du 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 du résultat :

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

- utiliser l'esperluette d'interprétation dans **SQL*Plus** pour restreindre et trier le résultat lors de l'exécution