ORACLE

3

Restricting and Sorting Data

Lesson Agenda

- Limiting rows
- Rules of precedence for operators in an expression
- Sorting rows using the ORDER BY clause
- SQL row limiting clause in a query
- Substitution variables in Oracle
- Assigning values to variables

Limiting Rows by Using a Selection

```
SELECT employee_id, last_name, job_id, department_id
FROM employees;
```

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90
4	103	Hunold	IT_PROG	60
5	104	Ernst	IT_PROG	60
6	107	Lorentz	IT_PROG	60

...

#	employee_id	last_name	job_id	department_id
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90
4	103	Hunold	IT_PROG	60
5	104	Ernst	IT_PROG	60
6	107	Lorentz	IT_PROG	60

...

What if you want to retrieve all employees in department 90, but not other departments?

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

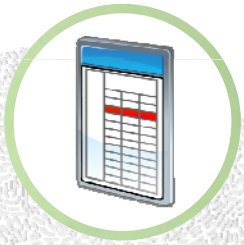
#	employee_id	last_name	job_id	department_id
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

Limiting Rows That Are Selected

- Restrict the rows that are returned by using the WHERE clause:

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

- The WHERE clause follows the FROM clause.



Using the WHERE Clause

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

	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90

#	employee_id	last_name	job_id	department_id
1	100	King	AD_PRES	90
2	101	Kochhar	AD_VP	90
3	102	De Haan	AD_VP	90
*	NULL	NULL	NULL	NULL

Character Strings and Dates

- Character strings and date values are enclosed within single quotation marks (`''`).
- Character values are case-sensitive and date values are format-sensitive.
- The default display format for date is `DD-MON-RR` in Oracle databases and `YYYY-MM-DD` in MySQL.

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

	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	Whalen	AD_ASST	10

#	last_name	job_id	department_id
1	Whalen	AD_ASST	10

```
SELECT last_name
FROM   employees
WHERE  hire_date = '17-OCT-11' ;
```

	LAST_NAME
1	Rajs

```
SELECT last_name, hire_date
FROM   employees
WHERE  hire_date = '2011-10-17' ;
```

#	last_name	hire_date
1	Rajs	2011-10-17

Comparison Operators

Operator	Meaning
=	Equal to
>	Greater than
>=	Greater than or equal to
<	Less than
<=	Less than or equal to
<>	Not equal to
BETWEEN ...AND...	Between two values (inclusive)
IN(set)	Match any of a list of values
LIKE	Match a character pattern
IS NULL	Is a null value

Using Comparison Operators

Let us look at some examples:

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

	LAST_NAME	SALARY
1	Matos	2600
2	Vargas	2500

#	last_name	salary
1	Matos	2600.00
2	Vargas	2500.00

```
SELECT *
FROM   employees
WHERE  last_name = 'Abel';
```

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	COMMISSION_PCT	MANAGER_ID	DEPARTMENT_ID
1	174	Ellen	Abel	EABEL	011.44.1644.429267	11-MAY-12	SA_REP	11000	0.3	149	80

#	employee_id	first_name	last_name	email	phone_numbe	hire_date	job_id	salary	commission_p	manager_id	department_id
1	174	Ellen	Abel	EABEL	011.44.1644....	2012-05-11	SA_REP	11000.00	0.30	149	80
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Range Conditions Using the BETWEEN Operator

You can use the BETWEEN operator to display rows based on a range of values:

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

Lower limit

Upper limit

	LAST_NAME	SALARY
1	Rajs	3500
2	Davies	3100
3	Matos	2600
4	Vargas	2500

#	last_name	salary
1	Rajs	3500.00
2	Davies	3100.00
3	Matos	2600.00
4	Vargas	2500.00

Using the IN Operator

Use the IN operator to test for values in a list:

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

	EMPLOYEE_ID	LAST_NAME	SALARY	MANAGER_ID
1	101	Kochhar	17000	100
2	102	De Haan	17000	100
3	124	Mourgos	5800	100
4	149	Zlotkey	10500	100
5	201	Hartstein	13000	100
6	200	Whalen	4400	101
7	205	Higgins	12008	101
8	202	Fay	6000	201

#	employee_id	last_name	salary	manager_id
1	101	Kochhar	17000.00	100
2	102	De Haan	17000.00	100
3	124	Mourgos	5800.00	100
4	149	Zlotkey	10500.00	100
5	201	Hartstein	13000.00	100
6	200	Whalen	4400.00	101
7	205	Higgins	12008.00	101
8	202	Fay	6000.00	201
*	NULL	NULL	NULL	NULL

Pattern Matching Using the LIKE Operator

- You can use the LIKE operator to perform wildcard searches of valid string patterns.
- The search conditions can contain either literal characters or numbers:
  - % denotes zero or more characters.
  - _ denotes one character.

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

	FIRST_NAME
1	Shelley
2	Steven

#	first_name
1	Shelley
2	Steven

Combining Wildcard Symbols

- You can combine the two wildcard symbols (%, _) with literal characters for pattern matching:

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

	LAST_NAME
1	Kochhar
2	Lorentz
3	Mourgos

#	last_name
1	Kochhar
2	Lorentz
3	Mourgos

- You can use the ESCAPE identifier to search for the actual % and _ symbols.

Using NULL Conditions

You can use the `IS NULL` operator to test for NULL values in a column.

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

	LAST_NAME	MANAGER_ID
1	King	(null)

#	last_name	manager_id
1	King	NULL

Defining Conditions Using Logical Operators

You can use the logical operators to filter the result set based on more than one condition or invert the result set.

Operator	Meaning
AND	Returns TRUE if both component conditions are true
OR	Returns TRUE if either component condition is true
NOT	Returns TRUE if the condition is false

Using the AND Operator

AND requires both the component conditions to be true:

```
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
1	149	Zlotkey	SA_MAN	10500
2	201	Hartstein	MK_MAN	13000

#	employee_id	last_name	job_id	salary
1	149	Zlotkey	SA_MAN	10500.00
2	201	Hartstein	MK_MAN	13000.00
*	NULL	NULL	NULL	NULL

Using the OR Operator

OR requires either component condition to be true:

```
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
1	100	King	AD_PRES	24000
2	101	Kochhar	AD_VP	17000
3	102	De Haan	AD_VP	17000
4	124	Mourgos	ST_MAN	5800
5	149	Zlotkey	SA_MAN	10500
6	174	Abel	SA_REP	11000
7	201	Hartstein	MK_MAN	13000
8	205	Higgins	AC_MGR	12008

#	employee_id	last_name	job_id	salary
1	100	King	AD_PRES	24000.00
2	101	Kochhar	AD_VP	17000.00
3	102	De Haan	AD_VP	17000.00
4	124	Mourgos	ST_MAN	5800.00
5	149	Zlotkey	SA_MAN	10500.00
6	174	Abel	SA_REP	11000.00
7	201	Hartstein	MK_MAN	13000.00
8	205	Higgins	AC_MGR	12008.00
*	NULL	NULL	NULL	NULL

Using the NOT Operator

NOT is used to negate a condition:

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

	LAST_NAME	JOB_ID
1	De Haan	AD_VP
2	Fay	MK_REP
3	Gietz	AC_ACCOUNT
4	Hartstein	MK_MAN
5	Higgins	AC_MGR
6	King	AD_PRES
7	Kochhar	AD_VP
8	Mourgos	ST_MAN
9	Whalen	AD_ASST
10	Zlotkey	SA_MAN

#	last_name	job_id
1	King	AD_PRES
2	Kochhar	AD_VP
3	De Haan	AD_VP
4	Mourgos	ST_MAN
5	Zlotkey	SA_MAN
6	Whalen	AD_ASST
7	Hartstein	MK_MAN
8	Fay	MK_REP
9	Higgins	AC_MGR
10	Gietz	AC_ACCOUNT

Lesson Agenda

- Limiting rows
- Rules of precedence for operators in an expression
- Sorting rows using the ORDER BY clause
- SQL row limiting clause in a query
- Substitution variables in Oracle
- Assigning values to variables

Rules of Precedence

Order	Operator
1	Arithmetic operators
2	Concatenation operator
3	Comparison conditions
4	`IS [NOT] NULL, LIKE, [NOT] IN`
5	`[NOT] BETWEEN`
6	Not equal to
7	`NOT` logical operator
8	`AND` logical operator
9	`OR` logical operator

You can use parentheses to override rules of precedence.

Rules of Precedence

1

```
SELECT last_name, department_id, salary
FROM   employees
WHERE  department_id = 60
OR     department_id = 80
AND    salary > 10000;
```

	LAST_NAME	DEPARTMENT_ID	SALARY
1	Hunold	60	9000
2	Ernst	60	6000
3	Lorentz	60	4200
4	Zlotkey	80	10500
5	Abel	80	11000

#	last_name	department_id	salary
1	Hunold	60	9000.00
2	Ernst	60	6000.00
3	Lorentz	60	4200.00
4	Zlotkey	80	10500.00
5	Abel	80	11000.00

2

```
SELECT last_name, department_id, salary
FROM   employees
WHERE  (department_id = 60
OR     department_id = 80)
AND    salary > 10000;
```

	LAST_NAME	DEPARTMENT_ID	SALARY
1	Zlotkey	80	10500
2	Abel	80	11000

#	last_name	department_id	salary
1	Zlotkey	80	10500.00
2	Abel	80	11000.00

Lesson Agenda

- Limiting rows
- Rules of precedence for operators in an expression
- Sorting rows using the ORDER BY clause
- SQL row limiting clause in a query
- Substitution variables in Oracle
- Assigning values to variables

Using the ORDER BY Clause

You can sort the retrieved rows with the ORDER BY clause:

- ASC: Ascending order, default
- DESC: Descending order

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

	LAST_NAME	JOB_ID	DEPARTMENT_ID	HIRE_DATE
1	De Haan	AD_VP	90	13-JAN-09
2	Kochhar	AD_VP	90	21-SEP-09
3	Higgins	AC_MGR	110	07-JUN-10
4	Gietz	AC_ACCOUNT	110	07-JUN-10
5	King	AD_PRES	90	17-JUN-11
6	Whalen	AD_ASST	10	17-SEP-11
7	Rajs	ST_CLERK	50	17-OCT-11

...

#	last_name	job_id	department_id	hire_date
1	De Haan	AD_VP	90	2009-01-13
2	Kochhar	AD_VP	90	2009-09-21
3	Gietz	AC_ACCOUNT	110	2010-06-07
4	Higgins	AC_MGR	110	2010-06-07
5	King	AD_PRES	90	2011-06-17
6	Whalen	AD_ASST	10	2011-09-17
7	Rajs	ST_CLERK	50	2011-10-17

...

Sorting

- Sorting in descending order:

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

1

- Sorting by column alias:

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

2

Sorting

- Sorting by using the column's numeric position:

```
SELECT   last_name, job_id, department_id, hire_date
FROM     employees
ORDER BY 3;
```

3

- Sorting by multiple columns:

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

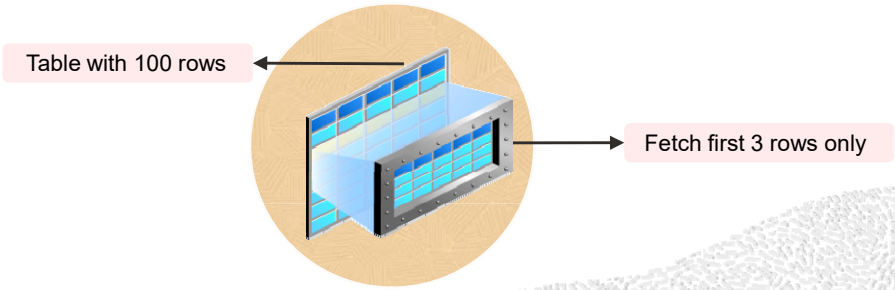
4

Lesson Agenda

- Limiting rows
- Rules of precedence for operators in an expression
- Sorting rows using the ORDER BY clause
- SQL row limiting clause in a query
- Substitution variables in Oracle
- Assigning values to variables

SQL Row Limiting Clause

- You can use the row limiting clause to limit the rows that are returned by a query.
- You can use this clause to implement Top-N reporting.



Using SQL Row Limiting Clause in a Query in Oracle

You specify `row_limiting_clause` in the SQL `SELECT` statement by placing it after the `ORDER BY` clause.

Syntax:

```
SELECT …
   FROM …
 [ WHERE …  ]
 [ ORDER BY …  ]
 [OFFSET offset { ROW | ROWS }]
[FETCH { FIRST | NEXT } [{ row_count | percent PERCENT }] { ROW
| ROWS }
  { ONLY | WITH TIES }]
```


SQL Row Limiting Clause: Example in Oracle

```
SELECT employee_id, first_name
FROM employees
ORDER BY employee_id
FETCH FIRST 5 ROWS ONLY;
```

Script Output | Query Result

SQL | All Rows Fetched: 5

	EMPLOYEE_ID	FIRST_NAME
1	100	Steven
2	101	Neena
3	102	Lex
4	103	Alexander
5	104	Bruce

```
SELECT employee_id, first_name
FROM employees
ORDER BY employee_id
OFFSET 5 ROWS FETCH NEXT 5 ROWS ONLY;
```

	EMPLOYEE_ID	FIRST_NAME
1	107	Diana
2	124	Kevin
3	141	Trenna
4	142	Curtis
5	143	Randall

Using SQL Row Limiting Clause in a Query in MySQL

You specify the row limiting clause in the SQL SELECT statement by placing it after the ORDER BY clause.

Syntax:

```
SELECT …
   FROM …
 [ WHERE …  ]
 [ ORDER BY …  ]
 [LIMIT {[offset,] row_count | row_count OFFSET offset}]
```


SQL Row Limiting Clause: Example in MySQL

```
SELECT employee_id, first_name
FROM employees
ORDER BY employee_id
LIMIT 7;
```

#	employee_id	first_name
1	100	Steven
2	101	Neena
3	102	Lex
4	103	Alexander
5	104	Bruce
6	107	Diana
7	124	Kevin
*	NULL	NULL

```
SELECT employee_id, first_name
FROM employees
ORDER BY employee_id
LIMIT 7 OFFSET 5;
```

#	employee_id	first_name
1	107	Diana
2	124	Kevin
3	141	Trenna
4	142	Curtis
5	143	Randall
6	144	Peter
7	149	Eleni
*	NULL	NULL

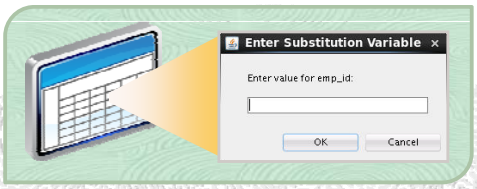
Lesson Agenda

- Limiting rows
- Rules of precedence for operators in an expression
- Sorting rows using the ORDER BY clause
- SQL row limiting clause in a query
- Substitution variables in Oracle
- Assigning values to variables

Substitution Variables in Oracle

Substitution Variables in Oracle

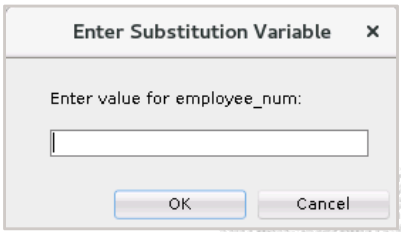
- Use substitution variables to:
  - Temporarily store values with single-ampersand (&) and double-ampersand (&&) substitution
- Use substitution variables to supplement:
  - WHERE conditions
  - ORDER BY clauses
  - Column expressions
  - Table names
  - Entire SELECT statements



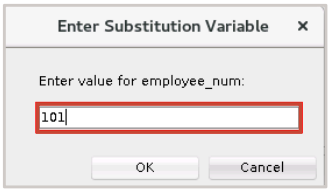
Using the Single-Ampersand Substitution Variable

Use a variable prefixed with an ampersand (&) to prompt the user for a value:

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



Using the Single-Ampersand Substitution Variable

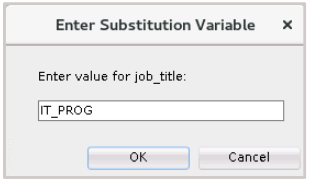


	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	101	Kochhar	17000	90

Character and Date Values with Substitution Variables

Use single quotation marks for date and character values:

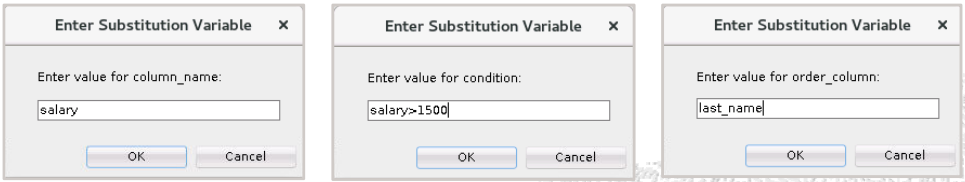
```
SELECT last_name, department_id, salary*12
FROM   employees
WHERE  job_id = '&job_title' ;
```



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

Specifying Column Names, Expressions, and Text

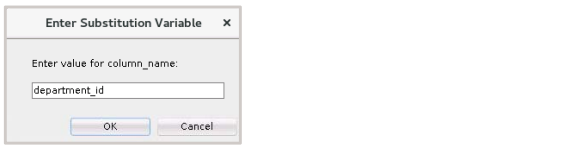
```
SELECT employee_id, last_name, job_id,&column_name
FROM   employees
WHERE  &condition
ORDER BY &order_column ;
```



Using the Double-Ampersand Substitution Variable

Use double ampersand (&&) if you want to reuse the variable value without prompting the user each time:

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



	EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
1	200	Whalen	AD_ASST	10
2	201	Hartstein	MK_MAN	20
3	202	Fay	MK_REP	20

...

Using the Ampersand Substitution Variable in SQL*Plus

```
oracle@[redacted]:~
File  Edit  View  Search  Terminal  Help
SQL> SELECT employee_id, last_name, salary, department_id
  2  from employees
  3  where employee_id = &employee_num;
Enter value for employee_num: 101
```

```
oracle@[redacted]
File  Edit  View  Search  Terminal  Help
SQL> SELECT employee_id, last_name, salary, department_id
  2  from employees
  3  where employee_id = &employee_num;
Enter value for employee_num: 101
old   3: where employee_id = &employee_num
new   3: where employee_id = 101

EMPLOYEE_ID LAST_NAME                     SALARY DEPARTMENT_ID
----------- ------------------------ ---------- -------------
        101 Kochhar                       17000            90

SQL>
```


Lesson Agenda

- Limiting rows
- Rules of precedence for operators in an expression
- Sorting rows using the ORDER BY clause
- SQL row limiting clause in a query
- Substitution variables in Oracle
- Assigning values to variables

Using the DEFINE Command in Oracle

- Use the DEFINE command to create a variable and assign a value to it.
- Use the UNDEFINE command to remove a variable.

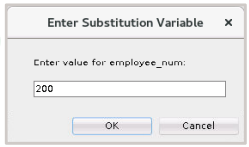
```
DEFINE employee_num = 200

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

UNDEFINE employee_num
```

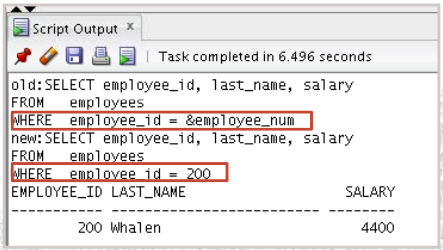
	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	200	Whalen	4400	10

Using the VERIFY Command in Oracle

Use the VERIFY command to toggle the display of the substitution variable, both before and after SQL Developer replaces substitution variables with values:

```
SET VERIFY ON
SELECT employee_id, last_name, salary
FROM   employees
WHERE  employee_id = &employee_num;
```





Using the SET Statement in MySQL

- Use the SET statement to create a user-defined variable and assign a value to it.
- User-defined variables are entered as *@var_name*.

```
SET @employee_num = 200;

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

#	employee_id	last_name	salary	department_id
1	200	Whalen	4400.00	10
*	NULL	NULL	NULL	NULL

Summary

In this lesson, you should have learned how to:

- Limit the rows that are retrieved by a query
- Sort the rows that are retrieved by a query