

4

Visualización de Datos de Varias Tablas

ORACLE

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Objetivos

Al finalizar esta lección, debería estar capacitado para:

- **Escribir sentencias `SELECT` para acceder a datos de más de una tabla utilizando uniones de igualdad y de no igualdad**
- **Visualizar datos que generalmente no cumplen una condición de unión utilizando uniones externas**
- **Unir una tabla consigo misma utilizando una autounión**

ORACLE

Objetivo de la Lección

Esta lección cubre el modo de obtener datos de más de una tabla.

Obtención de Datos de Varias Tablas

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

ORACLE

Datos de Varias Tablas

A veces es necesario utilizar datos de más de una tabla. En el ejemplo de la transparencia, el informe muestra datos de dos tablas distintas.

- Los identificadores de empleado están en la tabla EMPLOYEES.
- Los identificadores de departamento están en las tablas EMPLOYEES y DEPARTMENTS.
- Los identificadores de ubicación están en la tabla DEPARTMENTS.

Para producir el informe, debe enlazar las tablas EMPLOYEES y DEPARTMENTS y acceder a los datos de ambas.

Productos Cartesianos

- **Un producto Cartesiano se forma cuando:**
 - Una condición de unión está omitida.
 - Una condición de unión no es válida.
 - Todas las filas de la primera tabla se unen a todas las filas de la segunda tabla.
- **Para evitar un producto Cartesiano, incluya siempre una condición de unión válida en una cláusula `WHERE`.**

ORACLE

Productos Cartesianos

Cuando una condición de unión no es válida o está completamente omitida, el resultado es un *producto Cartesiano*, en el que se muestran todas las combinaciones de filas. Todas las filas de la primera tabla se unen con todas las filas de la segunda tabla.

Los productos Cartesianos tienden a generar un gran número de filas y es poco frecuente que el resultado sea útil. Debe incluir siempre una condición de unión válida en una cláusula `WHERE`, a menos que tenga la necesidad específica de combinar todas las filas de todas las tablas.

Los productos Cartesianos son útiles para algunas pruebas en las que se necesita generar un gran número de filas para simular una cantidad razonable de datos.

Generación de un Producto Cartesiano

EMPLOYEES (20 filas)

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 filas)

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.

**Producto
Cartesiano:** →
20x8=160 filas

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.

ORACLE

Productos Cartesianos (continuación)

Se genera un producto Cartesiano si una condición de unión está omitida. En el ejemplo de la transparencia se muestra el apellido del empleado y el nombre del departamento de las tablas EMPLOYEES y DEPARTMENTS. Como no se ha especificado ninguna cláusula WHERE, todas las filas (20) de la tabla EMPLOYEES se han unido con todas las filas (8) de la tabla DEPARTMENTS, generando así 160 filas en la salida.

```
SELECT last_name, department_name dept_name
FROM employees, departments;
```

LAST_NAME	DEPT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
...	

160 rows selected.

Tipos de Uniones

Uniones de Propiedad de Oracle (8i y anterior):

- Unión de igualdad
- Unión de no igualdad
- Unión externa
- Autounión

Uniones que cumplen con SQL: 1999:

- Uniones cruzadas
- Uniones naturales
- Cláusula USING
- Uniones externas completas o de dos lados
- Condiciones de unión arbitrarias para uniones externas

ORACLE

Tipos de Uniones

La base de datos Oracle9i ofrece una sintaxis de unión conforme con SQL: 1999. Antes de la versión 9i, la sintaxis de unión era distinta de los estándares ANSI. La nueva sintaxis de unión conforme con SQL: 1999 no ofrece ninguna ventaja de rendimiento con respecto a la sintaxis de unión de propiedad de Oracle que existía en las versiones anteriores.

Unión de Tablas Utilizando la Sintaxis Oracle

Utilice una unión para consultar datos de más de una tabla.

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

- Escriba la condición de unión en la cláusula WHERE.
- Escriba en el nombre de columna el nombre de tabla como prefijo si aparece el mismo nombre de columna en más de una tabla.

ORACLE

4-7

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Definición de Uniones

Cuando son necesarios datos de más de una tabla de la base de datos, se utiliza una condición de *unión*. Las filas de una tabla se pueden unir a las de otra según valores comunes que existen en las columnas correspondientes, es decir, normalmente columnas de clave primaria y ajena.

Para visualizar datos de dos o más tablas relacionadas, escriba una condición de unión simple en la cláusula WHERE.

En la sintaxis :

table1.column ndica la tabla y la columna de la que se recuperan los datos
table1.column1 = es la condición que une (o relaciona) las tablas entre sí
table2.column2

Instrucciones

- Al escribir una sentencia SELECT que una tablas, ponga delante del nombre de la columna el nombre de la tabla para obtener una mayor claridad y para mejorar el acceso a la base de datos.
- Si en más de una tabla aparece el mismo nombre de columna, éste debe llevar el nombre de tabla como prefijo.
- Para unir *n* tablas, necesita un mínimo de *n-1* condiciones de unión. Por ejemplo, para unir cuatro tablas, se requiere un mínimo de tres uniones. Esta regla puede no aplicarse si la tabla tiene una clave primaria concatenada, en cuyo caso se requiere más de una columna para identificar a cada fila de forma exclusiva.

Para obtener más información, consulte *Oracle9i SQL Reference*, "SELECT".

¿Qué Es una Unión de Igualdad?

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
50	Shipping
60	IT
60	IT
60	IT
80	Sales
80	Sales
80	Sales

...



Clave ajena Clave primaria

ORACLE

Uniones de Igualdad

Para determinar el nombre de departamento de un empleado, compare el valor de la columna DEPARTMENT_ID de la tabla EMPLOYEES con los valores de DEPARTMENT_ID de la tabla DEPARTMENTS. La relación entre las tablas EMPLOYEES y DEPARTMENTS es una *unión de igualdad*, es decir, los valores de la columna DEPARTMENT_ID de ambas tablas deben ser iguales. Con frecuencia, este tipo de unión implica complementos de clave primaria y ajena.

Nota: Las uniones de igualdad también se denominan *uniones simples* o *uniones internas*.

Recuperación de Registros con Uniones de Igualdad

```
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.

ORACLE

Recuperación de Registros con Uniones de Igualdad

En el ejemplo de la transparencia:

- La cláusula **SELECT** especifica los nombres de columna que se recuperan:
 - apellido del empleado, número del empleado y número de departamento, que son columnas de la tabla **EMPLOYEES**.
 - número de departamento, nombre de departamento e identificador de ubicación, que son columnas de la tabla **DEPARTMENTS**.
- La cláusula **FROM** especifica las dos tablas a las que debe acceder la base de datos:
 - tabla **EMPLOYEES**.
 - tabla **DEPARTMENTS**.
- La cláusula **WHERE** especifica cómo se deben unir las tablas:

EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID

Como la columna **DEPARTMENT_ID** es común a ambas tablas, debe tener como prefijo el nombre de la tabla para evitar ambigüedades.

Condiciones de Búsqueda Adicionales Utilizando el Operador 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
50	Shipping
60	IT
60	IT

...

ORACLE

4-10

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Condiciones de Búsqueda Adicionales

Además de la unión, puede tener criterios para la cláusula WHERE para restringir las filas en consideración para una o más tablas de la unión. Por ejemplo, para visualizar el número y el nombre de departamento del empleado Mato, necesita una condición adicional en la cláusula WHERE.

```
SELECT last_name, employees.department_id,
       department_name
FROM   employees, departments
WHERE  employees.department_id = departments.department_id
AND    last_name = 'Matos';
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Matos	50	Shipping

Cualificación de Nombres de Columna Ambiguos

- Utilice prefijos de tabla para cualificar nombres de columna que estén en varias tablas.
- Mejore el rendimiento utilizando prefijos de tabla.
- Distinga las columnas que tengan nombres idénticos pero que residan en tablas diferentes utilizando alias de columna.

ORACLE

4-11

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Cualificación de Nombres de Columna Ambiguos

Es necesario que cualifique los nombres de las columnas de la cláusula `WHERE` con el nombre de tabla para evitar ambigüedades. Sin los prefijos de tabla, la columna `DEPARTMENT_ID` podría ser de la tabla `DEPARTMENTS` o de la tabla `EMPLOYEES`. Es necesario agregar el prefijo de tabla para ejecutar la consulta.

Si no hay nombres de columna comunes entre dos tablas, no es necesario cualificar las columnas. Sin embargo, el uso del prefijo de tabla mejora el rendimiento ya que indica a Oracle Server dónde debe buscar exactamente las columnas.

La necesidad de cualificar los nombres de columna ambiguos también es aplicable a columnas que pueden ser ambiguas en otras cláusulas, como `SELECT` u `ORDER BY`.

Uso de Alias de Tabla

- Simplifique las consultas utilizando alias de tabla.
- Mejore el rendimiento utilizando prefijos de tabla.

```
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;
```

ORACLE

4-12

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Alias de Tabla

La cualificación de nombres de columna con nombres de tabla puede llevar mucho tiempo, especialmente si los nombres de tabla son largos. Puede utilizar en su lugar *alias de tabla*. Al igual que un alias de columna asigna a una columna otro nombre, un alias de tabla asigna a una tabla otro nombre. Los alias de tabla ayudan a reducir el código SQL, utilizando así menos memoria.

Observe cómo se identifican los alias de tabla en la cláusula FROM del ejemplo. El nombre de tabla se especifica completo, seguido de un espacio y del alias de tabla. A la tabla EMPLOYEES se le ha asignado el alias e y a la tabla DEPARTMENTS el alias d.

Instrucciones

- Los alias de tabla pueden tener hasta 30 caracteres, pero cuanto más pequeños sean, mejor.
- Si se utiliza un alias de tabla para un nombre de tabla en particular en la cláusula FROM, dicho alias se debe sustituir por el nombre de tabla en toda la sentencia SELECT.
- Los alias de tabla deben ser significativos.
- El alias de tabla sólo es válido para la sentencia SELECT actual.

Unión de Más de Dos Tablas

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

- Para unir n tablas, necesita un mínimo de $n-1$ condiciones de unión. Por ejemplo, para unir tres tablas, se requiere un mínimo de dos uniones.

ORACLE

Condiciones de Búsqueda Adicionales

A veces es posible que necesite unir más de dos tablas. Por ejemplo, para visualizar el apellido, el nombre de departamento y la ciudad de cada empleado, tiene que unir las tablas EMPLOYEES, DEPARTMENTS y LOCATIONS.

```
SELECT e.last_name, d.department_name, l.city
FROM   employees e, departments d, locations l
WHERE  e.department_id = d.department_id
AND    d.location_id = l.location_id;
```

LAST_NAME	DEPARTMENT_NAME	CITY
Hunold	IT	Southlake
Ernst	IT	Southlake
Lorentz	IT	Southlake
Mourgos	Shipping	South San Francisco
Rajs	Shipping	South San Francisco
Davies	Shipping	South San Francisco

■■■
19 rows selected.

Uniones de No Igualdad

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

← El salario de la tabla **EMPLOYEES** debe estar entre el salario menor y el mayor de la tabla **JOB_GRADES**.

ORACLE

Uniones de No Igualdad

Una unión de no igualdad es una condición de unión que contiene algo distinto a un operador de igualdad.

La relación entre las tablas **EMPLOYEES** y **JOB_GRADES** es un ejemplo de unión de no igualdad. Una relación entre las dos tablas es que la columna **SALARY** de la tabla **EMPLOYEES** debe estar entre los valores de las columnas **LOWEST_SALARY** y **HIGHEST_SALARY** de la tabla **JOB_GRADES**. La relación se obtiene utilizando un operador distinto de igual a (\neq).

Recuperación de Registros con Uniones de No Igualdad

```
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.

ORACLE

Uniones de No Igualdad (continuación)

El ejemplo de la transparencia crea una unión de no igualdad para evaluar el grado salarial del empleado. El salario debe estar *entre* cualquier par de rangos de salario bajo y alto.

Es importante observar que todos los empleados aparecen exactamente una vez en la lista cuando se ejecuta esta consulta y ninguno aparece repetido. Hay dos razones para ello:

- Ninguna de las filas de la tabla de grados de cargo contiene grados que se solapen, es decir, el valor de salario de un empleado solamente puede estar entre los valores de salario bajo y alto de una de las filas de la tabla de grados salariales.
- Todos los salarios de los empleados están dentro de los límites proporcionados por la tabla de grados de cargo, es decir, ningún empleado gana menos que el valor más bajo contenido en la columna `LOWEST_SAL` o más que el valor más alto contenido en la columna `HIGHEST_SAL`.

Nota: Se pueden utilizar otras condiciones, como `<=` and `>=`, pero `BETWEEN` es la más simple.

Recuerde especificar primero el valor bajo y después el alto al utilizar `BETWEEN`.

En el ejemplo de la transparencia se han especificado alias de tabla por motivos de rendimiento y no por una posible ambigüedad.

Uniones Externas

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.



No hay empleados en el departamento 190.

ORACLE

Devolución de Registros sin Coincidencia Directa con Uniones Externas

Si una fila no satisface una condición de unión, no aparecerá en el resultado de la búsqueda. Por ejemplo, en la condición de unión de igualdad de las tablas EMPLOYEES y DEPARTMENTS, el empleado Grant no aparece porque no hay identificador de departamento registrado para él en la tabla EMPLOYEES. En lugar de ver 20 empleados en el juego de resultados, ve 19.

```
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

19 rows selected.

Sintaxis de Uniones Externas

- También puede utilizar una unión externa para ver filas que no cumplen la condición de unión.
- El operador de unión externa es el signo más (+).

```
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(+);
```

ORACLE

Uso de Uniones Externas para Devolver Registros sin Coincidencia Directa

Las filas que faltan se pueden devolver si se utiliza un operador de *unión externa* en la condición de unión. El operador es un signo más entre paréntesis (+) y *se coloca “al lado” de la unión que tiene información insuficiente*. Este operador tiene el efecto de crear una o varias filas nulas, a las que se pueden unir una o varias filas de la tabla que no tiene información insuficiente.

En la sintaxis:

<code>table1.column =</code>	es la condición que une (o relaciona) las tablas entre sí.
<code>table2.column (+)</code>	es el símbolo de unión externa, que se puede colocar en cualquier lado de la condición de cláusula WHERE, pero no en ambos. (Coloque el símbolo de unión externa a continuación del nombre de la columna en la tabla sin filas coincidentes.)

Uso de Uniones Externas

```
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.

ORACLE

Uso de Uniones Externas para Devolver Registros sin Coincidencia Directa (continuación)

En el ejemplo de la transparencia se muestran los apellidos de los empleados, los identificadores de departamento y los nombres de departamento. El departamento Contracting no tiene empleados. En la salida mostrada aparece el valor vacío.

Restricciones de Unión Externa

- El operador de unión externa puede aparecer solamente en *un* lado de la expresión, el lado en el que falta información. Devuelve las filas de una tabla que no tienen coincidencia directa en la otra tabla.
- Una condición que implique una unión externa no puede utilizar el operador IN ni estar enlazada a otra condición con el operador OR.

Autouniones

EMPLOYEES (WORKER)

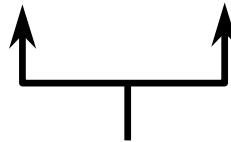
EMPLOYEE_ID	LAST_NAME	MANAGER_ID
100	King	
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103
107	Lorentz	103
124	Mourgos	100

...

EMPLOYEES (MANAGER)

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

...



**MANAGER_ID en la tabla WORKER es igual a
EMPLOYEE_ID en la tabla MANAGER.**

ORACLE

Unión de una Tabla Consigo Misma

A veces, es necesario unir una tabla consigo misma. Para buscar el nombre del director de cada empleado, necesita unir la tabla EMPLOYEES consigo misma o realizar una autounión. Por ejemplo, para buscar el nombre del director de Whalen, necesita:

- Buscar a Whalen en la tabla EMPLOYEES mirando en la columna LAST_NAME.
- Buscar el número de director para Whalen mirando en la columna MANAGER_ID. El número del director de Whalen es 101.
- Buscar el nombre del director con EMPLOYEE_ID 101 mirando en la columna LAST_NAME. El número del empleado Kochhar es 101, por lo que Kochhar es el director de Whalen.

En este proceso, tiene que mirar dos veces en la tabla. La primera vez para buscar a Whalen en la columna LAST_NAME y el valor 101 de MANAGER_ID. La segunda vez mira la columna EMPLOYEE_ID para buscar 101 y la columna LAST_NAME para buscar a Kochhar.

Unión de una Tabla Consigo Misma

```
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.

ORACLE

Unión de una Tabla Consigo Misma (continuación)

El ejemplo de la transparencia une a la tabla EMPLOYEES consigo misma. Para simular dos tablas en la cláusula FROM, hay dos alias, w y m, para la misma tabla, EMPLOYEES.

En este ejemplo, la cláusula WHERE contiene la unión que significa “donde el número de director de un trabajador coincide con el número de empleado para el director”.

Práctica 4, Parte Uno: Visión General

Esta práctica cubre la escritura de consultas para unir tablas utilizando la sintaxis Oracle.

ORACLE

4-21

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Práctica 4, Parte Uno

Esta práctica se ha diseñado para ofrecerle diversos ejercicios que unen tablas utilizando la sintaxis Oracle mostrada hasta ahora en la lección.

Conteste a las preguntas de práctica 1- 4 al final de esta lección.

Unión de Tablas Utilizando la Sintaxis SQL: 1999

Utilice una unión para consultar datos de más de una tabla.

```
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)];
```

ORACLE

4-22

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Definición de Uniones

Al utilizar la sintaxis SQL: 1999, puede obtener resultados iguales a los mostrados en páginas anteriores.

En la sintaxis:

<i>table1.column</i>	Indica la tabla y la columna de la que se recuperan los datos.
CROSS JOIN	Devuelve un producto Cartesiano de las dos tablas.
NATURAL JOIN	Une dos tablas basadas en el mismo nombre de columna.
JOIN <i>table</i>	
USING <i>column_name</i>	Realiza una unión de igualdad basada en el nombre de columna
JOIN <i>table</i> ON	
<i>table1.column_name</i>	Realiza una unión de igualdad basada en la condición de la cláusula ON
= <i>table2.column_name</i>	
LEFT/RIGHT/FULL OUTER	

Para obtener más información, consulte *Oracle9i SQL Reference*, “SELECT”.

Creación de Uniones Cruzadas

- La cláusula **CROSS JOIN** produce varios productos entre dos tablas.
- Es lo mismo que un producto Cartesiano entre las dos tablas.

```
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.

ORACLE

Creación de Uniones Cruzadas

El ejemplo de la transparencia ofrece resultados iguales a los siguientes:

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

LAST_NAME	DEPARTMENT_NAME
King	Administration
Kochhar	Administration
De Haan	Administration
Hunold	Administration
Ernst	Administration

■■■
160 rows selected.

Creación de Uniones Naturales

- La cláusula `NATURAL JOIN` se basa en todas las columnas de las dos tablas que tienen el mismo nombre.
- Selecciona filas de las dos tablas que tienen los mismos valores en todas las columnas coincidentes.
- Si las columnas que tienen el mismo nombre tienen distintos tipos de dato, se devuelve un error.

ORACLE

Creación de Uniones Naturales

En las versiones anteriores de Oracle, no era posible realizar una unión sin especificar de forma explícita las columnas en las tablas correspondientes. En Oracle9i se puede dejar que se termine automáticamente la unión basada en columnas de las dos tablas que tienen tipos de dato y nombres coincidentes, utilizando las palabras clave `NATURAL JOIN`.

Nota: La unión sólo puede ocurrir en columnas que tengan los mismos nombres y tipos de dato en las dos tablas. Si las columnas tienen el mismo nombre, pero distintos tipos de dato, la sintaxis `NATURAL JOIN` produce un error.

Recuperación de Registros con Uniones Naturales

```
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.

ORACLE

Recuperación de Registros con Uniones Naturales

En el ejemplo de la transparencia, la tabla LOCATIONS está unida a la tabla DEPARTMENT por la columna LOCATION_ID, que es la única columna con el mismo nombre en las dos tablas. Si estuvieran presentes otras columnas comunes, la unión las habría utilizado a todas.

Uniones de Igualdad

La unión natural también se puede escribir como unión de igualdad:

```
SELECT department_id, department_name,  
       departments.location_id, city  
FROM   departments, locations  
WHERE  departments.location_id = locations.location_id;
```

Uniones Naturales con una Cláusula WHERE

Con la cláusula WHERE se implementan restricciones adicionales sobre una unión natural. El ejemplo siguiente limita las filas de salida a las que tienen un identificador de departamento igual a 20 ó 50.

```
SELECT department_id, department_name,  
       location_id, city  
FROM   departments  
NATURAL JOIN locations  
WHERE  department_id IN (20, 50);
```

Creación de Uniones con la Cláusula USING

- Si varias columnas tienen los mismos nombres pero los tipos de dato no coinciden, la cláusula `NATURAL JOIN` se puede modificar con la cláusula `USING` para especificar las columnas que se deben utilizar para una unión de igualdad.
- Utilice la cláusula `USING` para hacer coincidir solamente una columna cuando coincidan varias.
- No utilice un nombre o alias de tabla en las columnas de referencia.
- Las cláusulas `NATURAL JOIN` y `USING` son mutuamente excluyentes.

ORACLE

4-26

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

La Cláusula USING

Las uniones naturales utilizan todas las columnas con nombres y tipos de dato coincidentes para unir las tablas. La cláusula `USING` se puede utilizar para especificar solamente las columnas que se deben utilizar para una unión de igualdad. Las columnas de referencia en la cláusula `USING` no deben tener un cualificador (nombre o alias de tabla) en ningún lugar de la sentencia SQL.

Por ejemplo, esta sentencia es válida:

```
SELECT l.city, d.department_name
FROM   locations l JOIN departments d USING (location_id)
WHERE  location_id = 1400;
```

Esta sentencia no es válida porque `LOCATION_ID` está cualificado en la cláusula `WHERE`:

```
SELECT l.city, d.department_name
FROM   locations l JOIN departments d USING (location_id)
WHERE  d.location_id = 1400;
ORA-25154: column part of USING clause cannot have qualifier
```

La misma restricción se aplica también a las uniones `NATURAL`. Por lo tanto, las columnas que tienen el mismo nombre en las dos tablas se deben utilizar sin ningún cualificador.

Recuperación de Registros con la Cláusula 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.

ORACLE

La Cláusula USING (continuación)

El ejemplo mostrado une la columna DEPARTMENT_ID de las tablas EMPLOYEES y DEPARTMENTS y muestra de este modo la ubicación en la que trabaja un empleado.

También se puede escribir como unión de igualdad:

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

Creación de Uniones con la Cláusula ON

- La condición de unión para la unión natural es básicamente una unión de igualdad de todas las columnas con el mismo nombre.
- Para especificar condiciones arbitrarias o especificar columnas para unir, se utiliza la cláusula ON.
- La condición de unión se separa de otras condiciones de *búsqueda*.
- La cláusula ON facilita la comprensión del código.

ORACLE

La Condición ON

Utilice la cláusula ON para especificar una condición de unión. Esto le permite especificar condiciones de unión distintas de cualquier condición de búsqueda o filtro en la cláusula WHERE.

Recuperación de Registros con la Cláusula 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.

ORACLE

4-29

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Creación de Uniones con la Cláusula ON

La cláusula ON también se puede utilizar como se indica a continuación para unir columnas que tienen nombres distintos:

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

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

...

19 rows selected.

El ejemplo anterior es una autounión de la tabla EMPLOYEE consigo misma, basada en las columnas EMPLOYEE_ID y MANAGER_ID.

Creación de Uniones en Tres Sentidos con la Cláusula 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.

ORACLE

Uniones en Tres Sentidos

Una unión en tres sentidos es una unión de tres tablas. En la sintaxis conforme con SQL: 1999, las uniones se realizan de izquierda a derecha, por lo que la primera que se realiza es EMPLOYEES JOIN DEPARTMENTS. La primera condición de unión puede hacer referencia a columnas de EMPLOYEES y de DEPARTMENTS, pero no a columnas de referencia en LOCATIONS. La segunda condición de unión puede hacer referencia a columnas de las tres tablas.

También se puede escribir como unión de igualdad en tres sentidos:

```
SELECT employee_id, city, department_name
FROM   employees, departments, locations
WHERE  employees.department_id = departments.department_id
AND    departments.location_id = locations.location_id;
```

Uniones INNER frente a OUTER

- En SQL: 1999, la unión de dos tablas que devuelve solamente las filas coincidentes es una unión interna.
- Una unión entre dos tablas que devuelve los resultados de la unión interna así como las tablas izquierda (o derecha) de filas no coincidentes es una unión externa izquierda (o derecha).
- Una unión entre dos tablas que devuelve los resultados de una unión interna así como los de una unión izquierda y derecha es una unión externa completa.

ORACLE

Uniones: Comparación de la Sintaxis SQL : 1999 con la Sintaxis Oracle

Oracle	SQL: 1999
Unión de igualdad	Unión natural/interna
Unión externa	Unión externa izquierda
Autounión	Unión ON
Unión de no igualdad	Unión USING
Producto Cartesiano	Unión cruzada

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.

ORACLE

Ejemplo de LEFT OUTER JOIN

Esta consulta recupera todas las filas de la tabla EMPLOYEES, que es la tabla izquierda aunque no haya ninguna coincidencia en la tabla DEPARTMENTS.

En versiones anteriores esta consulta se terminó como se indica a continuación:

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


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.

ORACLE

Ejemplo de RIGHT OUTER JOIN

Esta consulta recupera todas las filas de la tabla DEPARTMENTS, que es la tabla derecha aunque no haya ninguna coincidencia en la tabla EMPLOYEES.

En versiones anteriores esta consulta se terminó como se indica a continuación:

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

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.

ORACLE

Ejemplo de FULL OUTER JOIN

Esta consulta recupera todas las filas de la tabla EMPLOYEES, aunque no haya ninguna coincidencia en la tabla DEPARTMENTS. También recupera todas las filas de la tabla DEPARTMENTS, aunque no haya ninguna coincidencia en la tabla EMPLOYEES.

Condiciones Adicionales

```
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

ORACLE

Aplicación de Condiciones Adicionales

Puede aplicar condiciones adicionales en la cláusula WHERE. En el ejemplo mostrado se realiza una unión en las tablas EMPLOYEES y DEPARTMENTS y, además, solamente se visualizan los empleados con un identificador de director igual a 149.

Resumen

En esta lección, debería haber aprendido a utilizar uniones para visualizar datos de varias tablas en:

- **Sintaxis de propiedad de Oracle para las versiones 8i y anteriores**
- **Sintaxis conforme con SQL: 1999 para la versión 9i**

ORACLE

4-36

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Resumen

Hay varias formas de unir tablas.

Tipos de Uniones

- Uniones de igualdad
- Uniones de no igualdad
- Uniones externas
- Autouniones
- Uniones cruzadas
- Uniones naturales
- Uniones completas o externas

Productos Cartesianos

Un producto Cartesiano da como resultado el despliegue de todas las combinaciones de filas. Para ello, se omite la cláusula `WHERE` o se especifica la cláusula `CROSS JOIN`.

Alias de Tabla

- Los alias de tabla aceleran el acceso a la base de datos.
- Los alias de tabla pueden ayudar a reducir el código SQL, conservando la memoria.

Práctica 4, Parte Dos: Visión General

Esta práctica cubre los siguientes temas:

- **Unión de tablas utilizando una unión de igualdad**
- **Realización de uniones externas y autouniones**
- **Adición de condiciones**

ORACLE

4-37

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Práctica 4, Parte Dos

Esta práctica se ha diseñado para proporcionarle experiencia práctica en la extracción de datos de más de una tabla. Intente utilizar tanto la sintaxis de propiedad de Oracle como la conforme con SQL: 1999.

En las preguntas 5-8 de la Parte Dos, intente escribir las sentencias de unión utilizando la sintaxis ANSI.

En las preguntas 9-11 de la Parte Dos, intente escribir las sentencias de unión utilizando tanto la sintaxis Oracle como ANSI.

Práctica 4, Parte Uno

1. Escriba una consulta para visualizar el apellido del empleado, el número y el nombre de departamento para todos los empleados.

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
Vargas	50	Shipping
Hunold	60	IT
Ernst	60	IT
Lorentz	60	IT
Zlotkey	80	Sales
Abel	80	Sales

■ ■ ■

19 rows selected.

2. Cree un listado único de todos los cargos que haya en el departamento 80. Incluya la ubicación del departamento en el resultado.

JOB_ID	LOCATION_ID
SA_MAN	2500
SA_REP	2500

3. Escriba una consulta para mostrar el apellido del empleado, el nombre de departamento, el identificador de ubicación y la ciudad de todos los empleados que perciben comisión.

LAST_NAME	DEPARTMENT_NAME	LOCATION_ID	CITY
Zlotkey	Sales	2500	Oxford
Abel	Sales	2500	Oxford
Taylor	Sales	2500	Oxford

Práctica 4, Parte Uno (continuación)

- Visualice el apellido del empleado y el nombre de departamento para todos los empleados que tengan una *a* (minúsculas) en el apellido. Coloque la sentencia SQL en un archivo de texto llamado `lab4_4.sql`.

LAST_NAME	DEPARTMENT_NAME
Whalen	Administration
Hartstein	Marketing
Fay	Marketing
Rajs	Shipping
Davies	Shipping
Matos	Shipping
Vargas	Shipping
Taylor	Sales
Kochhar	Executive
De Haan	Executive

10 rows selected.

Práctica 4, Parte Dos

5. Escriba una consulta para visualizar el apellido, el cargo, el número y el nombre de departamento para todos los empleados que trabajan en Toronto.

LAST_NAME	JOB_ID	DEPARTMENT_ID	DEPARTMENT_NAME
Hartstein	MK_MAN	20	Marketing
Fay	MK_REP	20	Marketing

6. Visualice el apellido y el número del empleado junto con el apellido y el número de su director. Etiquete las columnas como Employee, Emp#, Manager y Mgr#, respectivamente. Coloque la sentencia SQL en un archivo de texto llamado lab4_6.sql.

Employee	EMP#	Manager	Mgr#
Kochhar	101	King	100
De Haan	102	King	100
Mourgos	124	King	100
Zlotkey	149	King	100
Hartstein	201	King	100
Whalen	200	Kochhar	101
Higgins	205	Kochhar	101
Hunold	103	De Haan	102
Ernst	104	Hunold	103
Lorentz	107	Hunold	103
Rajs	141	Mourgos	124
Davies	142	Mourgos	124
Matos	143	Mourgos	124
Vargas	144	Mourgos	124
Employee	EMP#	Manager	Mgr#
Abel	174	Zlotkey	149
Taylor	176	Zlotkey	149
Grant	178	Zlotkey	149
Fay	202	Hartstein	201
Gietz	206	Higgins	205

19 rows selected.

Práctica 4, Parte Dos (continuación)

7. Modifique `lab4_6.sql` para visualizar a todos los empleados incluyendo a King, que no tiene director. Ordene los resultados por número de empleado.
Coloque la sentencia SQL en un archivo de texto llamado `lab4_7.sql`. Ejecute la consulta en `lab4_7.sql`.

Employee	EMP#	Manager	Mgr#
King	100		
Kochhar	101	King	100
De Haan	102	King	100
Hunold	103	De Haan	102
Ernst	104	Hunold	103
Lorentz	107	Hunold	103
Mourgos	124	King	100

■ ■ ■

20 rows selected.

Si tiene tiempo, complete los siguientes ejercicios:

8. Cree una consulta que muestre apellidos de empleado, números de departamento y todos los empleados que trabajan en el mismo departamento que un empleado dado. Asigne a cada columna la etiqueta adecuada.

DEPARTMENT	EMPLOYEE	COLLEAGUE
20	Fay	Hartstein
20	Hartstein	Fay
50	Davies	Matos
50	Davies	Mourgos
50	Davies	Rajs
50	Davies	Vargas
50	Matos	Davies
50	Matos	Mourgos
50	Matos	Rajs
50	Matos	Vargas
50	Mourgos	Davies
50	Mourgos	Matos
50	Mourgos	Rajs
50	Mourgos	Vargas

■ ■ ■

42 rows selected.

Práctica 4, Parte Dos (continuación)

9. Visualice la estructura de la tabla `JOB_GRADES`. Cree una consulta en la que pueda visualizar el nombre, el cargo, el nombre de departamento, el salario y el grado de todos los empleados.

Name	Null?	Type
GRADE_LEVEL		VARCHAR2(3)
LOWEST_SAL		NUMBER
HIGHEST_SAL		NUMBER

LAST_NAME	JOB_ID	DEPARTMENT_NAME	SALARY	GRA
Matos	ST_CLERK	Shipping	2600	A
Vargas	ST_CLERK	Shipping	2500	A
Lorentz	IT_PROG	IT	4200	B
Mourgos	ST_MAN	Shipping	5800	B
Rajs	ST_CLERK	Shipping	3500	B
Davies	ST_CLERK	Shipping	3100	B
Whalen	AD_ASST	Administration	4400	B

■ ■ ■

19 rows selected.

Si desea una comprobación adicional, complete los siguientes ejercicios:

10. Cree una consulta para visualizar el apellido y la fecha de contratación de cualquier empleado contratado después del empleado Davies.

LAST_NAME	HIRE_DATE
Lorentz	07-FEB-99
Mourgos	16-NOV-99
Matos	15-MAR-98
Vargas	09-JUL-98
Zlotkey	29-JAN-00
Taylor	24-MAR-98
Grant	24-MAY-99
Fay	17-AUG-97

8 rows selected.

Práctica 4, Parte Dos (continuación)

11. Visualice los nombres y las fechas de contratación de todos los empleados contratados antes que sus directores, junto con los nombres y las fechas de contratación de estos últimos. Etiquete las columnas como Employee, Emp Hired, Manager y Mgr Hired, respectivamente.

LAST_NAME	HIRE_DATE	LAST_NAME	HIRE_DATE
Whalen	17-SEP-87	Kochhar	21-SEP-89
Hunold	03-JAN-90	De Haan	13-JAN-93
Rajs	17-OCT-95	Mourgos	16-NOV-99
Davies	29-JAN-97	Mourgos	16-NOV-99
Matos	15-MAR-98	Mourgos	16-NOV-99
Vargas	09-JUL-98	Mourgos	16-NOV-99
Abel	11-MAY-96	Zlotkey	29-JAN-00
Taylor	24-MAR-98	Zlotkey	29-JAN-00
Grant	24-MAY-99	Zlotkey	29-JAN-00

9 rows selected.

5

Agregado de Datos Utilizando Funciones de Grupo

ORACLE

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Objetivos

Al finalizar esta lección, debería estar capacitado para:

- **Identificar las funciones de grupo disponibles**
- **Describir el uso de las funciones de grupo**
- **Agrupar datos utilizando la cláusula `GROUP BY`**
- **Incluir o excluir filas agrupadas utilizando la cláusula `HAVING`**

ORACLE

5-2

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Objetivo de la Lección

Esta lección vuelve a tratar sobre las funciones. Se centra en la obtención de información de resumen, como medias, para grupos de filas. También trata sobre cómo agrupar las filas de una tabla en juegos más pequeños y cómo especificar los criterios de búsqueda para grupos de filas.

¿Qué Son Funciones de Grupo?

Las funciones de grupo operan sobre juegos de filas para proporcionar un resultado por grupo.

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.

El salario
máximo en
la tabla
EMPLOYEES.

MAX(SALARY)
24000

ORACLE

Funciones de Grupo

A diferencia de las funciones de una sola fila, las funciones de grupo operan sobre juegos de filas para proporcionar un resultado por grupo. Estos juegos pueden ser la tabla completa o la tabla dividida en grupos.

Tipos de Funciones de Grupo

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

ORACLE

5-4

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Funciones de Grupo (continuación)

Cada una de las funciones acepta un argumento. En la siguiente tabla se identifican las opciones que puede utilizar en la sintaxis:

Función	Descripción
AVG([DISTINCT ALL] <i>n</i>)	Valor medio de <i>n</i> , ignorando los valores nulos
COUNT({ * [DISTINCT ALL] <i>expr</i> })	Número de filas, donde <i>expr</i> se evalúa en algo distinto de nulo (cuenta todas las filas seleccionadas utilizando *, incluyendo duplicados y filas con nulos)
MAX([DISTINCT ALL] <i>expr</i>)	Valor máximo de <i>expr</i> , ignorando los valores nulos
MIN([DISTINCT ALL] <i>expr</i>)	Valor mínimo de <i>expr</i> , ignorando los valores nulos
STDDEV([DISTINCT ALL] <i>x</i>)	Desviación estándar de <i>n</i> , ignorando los valores nulos
SUM([DISTINCT ALL] <i>n</i>)	Suma de los valores de <i>n</i> , ignorando los valores nulos
VARIANCE([DISTINCT ALL] <i>x</i>)	Varianza de <i>n</i> , ignorando los valores nulos

Sintaxis de las Funciones de Grupo

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

ORACLE

5-5

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Instrucciones para el Uso de Funciones de Grupo

- DISTINCT hace que la función solamente considere valores no duplicados; ALL hace que considere todos los valores incluyendo duplicados. El valor por defecto es ALL y, por lo tanto, no es necesario especificarlo.
- Los tipos de dato para las funciones con un argumento *expr* pueden ser CHAR, VARCHAR2, NUMBER o DATE.
- Todas las funciones de grupo ignoran los valores nulos. Para sustituir un valor por valores nulos, utilice las funciones NVL, NVL2 o COALESCE.
- Oracle Server ordena implícitamente el juego de resultados en orden ascendente al utilizar una cláusula GROUP BY. Para sustituir este orden por defecto, se puede utilizar DESC en una cláusula ORDER BY.

Uso de las Funciones AVG y SUM

Puede utilizar AVG y SUM para datos numéricos.

```
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

ORACLE

Funciones de Grupo

Puede utilizar las funciones AVG, SUM, MIN y MAX en columnas que almacenan datos numéricos. En el ejemplo de la transparencia se muestra la media, el salario mayor, el menor y la suma de los salarios mensuales de todos los representantes de ventas.

Uso de las Funciones MIN y MAX

Puede utilizar MIN y MAX para cualquier tipo de dato.

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

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

ORACLE

5-7

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Funciones de Grupo (continuación)

Puede utilizar las funciones MAX y MIN para cualquier tipo de dato. En el ejemplo de la transparencia se muestra al empleado de menor edad y al de mayor edad.

En el siguiente ejemplo se muestra el apellido del primer y del último empleado de una lista alfabética de todos los empleados.

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

MIN(LAST_NAME)	MAX(LAST_NAME)
Abel	Zlotkey

Nota: Las funciones AVG, SUM, VARIANCE y STDDEV sólo se pueden utilizar con tipos de dato numéricos.

Uso de la Función COUNT

COUNT (*) devuelve el número de filas de una tabla.

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

COUNT(*)
5

ORACLE

5-8

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

La Función COUNT

La función COUNT tiene tres formatos:

- COUNT (*)
- COUNT (*expr*)
- COUNT (DISTINCT *expr*)

COUNT (*) devuelve el número de filas de una tabla que satisface los criterios de la sentencia SELECT, incluyendo filas duplicadas y filas que contengan valores nulos en cualquiera de las columnas. Si se incluye una cláusula WHERE en la sentencia SELECT, COUNT (*) devuelve el número de filas que satisface la condición de la cláusula WHERE..

En contraste, COUNT (*expr*) devuelve el número de valores no nulos de la columna identificada por *expr*.

COUNT (DISTINCT *expr*) devuelve el número de valores únicos no nulos de la columna identificada por *expr*.

En el ejemplo de la transparencia se muestra el número de empleados del departamento 50.

Uso de la Función COUNT

- `COUNT(expr)` devuelve el número de filas con valores no nulos para `expr`.
- Visualice el número de valores de departamento de la tabla `EMPLOYEES`, excluyendo los valores nulos.

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

COUNT(COMMISSION_PCT)
3

ORACLE

5-9

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

La Función COUNT (continuación)

En el ejemplo de la transparencia se muestra el número de empleados del departamento 80 que pueden percibir una comisión.

Ejemplo

Visualice el número de valores de departamento de la tabla `EMPLOYEES`.

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

COUNT(DEPARTMENT_ID)
19

Uso de la Palabra Clave DISTINCT

- **COUNT(DISTINCT expr)** devuelve el número de valores distintos no nulos de *expr*.
- Visualice el número de valores de departamento distintos de la tabla **EMPLOYEES**.

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

COUNT(DISTINCTDEPARTMENT_ID)
7

ORACLE

La Palabra Clave DISTINCT

Utilice la palabra clave **DISTINCT** para suprimir el recuento de cualquier valor duplicado dentro de una columna.

En el ejemplo de la transparencia se muestra el número de valores de departamento distintos de la tabla **EMPLOYEES**.

Funciones de Grupo y Valores Nulos

Las funciones de grupo ignoran los valores nulos de la columna.

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

AVG(COMMISSION_PCT)
.2125

ORACLE

Funciones de Grupo y Valores Nulos

Todas las funciones de grupo ignoran los valores nulos de la columna. En el ejemplo de la transparencia, la media se calcula basándose *solamente* en las filas de la tabla que almacenan un valor válido en la columna COMMISSION_PCT. La media se calcula como la comisión total pagada a todos los empleados dividida por el número de empleados que perciben una comisión (cuatro).

Uso de la Función NVL con Funciones de Grupo

La función NVL fuerza a las funciones de grupo a que incluyan valores nulos.

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

AVG(NVL(COMMISSION_PCT,0))	
	.0425

ORACLE

Funciones de Grupo y Valores Nulos (continuación)

La función NVL fuerza a que las funciones de grupo incluyan valores nulos. En el ejemplo de la transparencia, la media se calcula basándose en *todas* las filas de la tabla, independientemente de si se almacenan valores nulos en la columna COMMISSION_PCT. La media se calcula como la comisión total pagada a todos los empleados dividida por el número total de empleados de la compañía (20).

Creación de Grupos de Datos

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

El
salario
medio
de la
tabla
EMPLOYEES
para cada
departamento.

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

ORACLE

Grupos de Datos

Hasta ahora, todas las funciones de grupo han tratado la tabla como un gran grupo de información. A veces, necesita dividir la tabla de información en grupos más pequeños. Esto se puede realizar utilizando la cláusula GROUP BY.

Creación de Grupos de Datos: Sintaxis de la Cláusula GROUP BY

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

Divida las filas de una tabla en grupos más pequeños utilizando la cláusula GROUP BY.

ORACLE

5-14

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

La Cláusula GROUP BY

Puede utilizar la cláusula GROUP BY para dividir las filas de una tabla en grupos. A continuación, puede utilizar las funciones de grupo para devolver información de resumen para cada grupo.

En la sintaxis:

group_by_expression especifica columnas cuyos valores determinan la base para agrupar filas

Instrucciones

- Si incluye una función de grupo en una cláusula SELECT, no puede seleccionar también resultados individuales, *a menos que* la columna individual aparezca en la cláusula GROUP BY. Recibirá un mensaje de error si no incluye la lista de columnas en la cláusula GROUP BY.
- Si utiliza una cláusula WHERE, puede excluir filas antes de dividir las en grupos.
- Debe incluir las *columnas* en la cláusula GROUP BY.
- No se puede utilizar un alias de columna en la cláusula GROUP BY.
- Por defecto, las filas se ordenan en orden ascendente de las columnas incluidas en la lista GROUP BY. Puede sustituir este orden por defecto utilizando la cláusula ORDER BY.

Uso de la Cláusula GROUP BY

Todas las columnas de la lista **SELECT** que no estén en las funciones de grupo deben estar en la cláusula **GROUP BY**.

```
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.

ORACLE

La Cláusula GROUP BY (continuación)

Al utilizar la cláusula **GROUP BY**, asegúrese de que todas las columnas de la lista **SELECT** que no son funciones de grupo están incluidas en la cláusula **GROUP BY**. En el ejemplo de la transparencia se muestra el número de departamento y el salario medio para cada departamento. A continuación, se muestra el modo en que se evalúa esta sentencia **SELECT**, que contiene una cláusula **GROUP BY**:

- La cláusula **SELECT** especifica las columnas que se van a recuperar:
 - Columna de número de departamento de la tabla **EMPLOYEES**
 - La media de todos los salarios en el grupo que ha especificado en la cláusula **GROUP BY**
- La cláusula **FROM** especifica las tablas a las que debe acceder la base de datos: la tabla **EMPLOYEES**.
- La cláusula **WHERE** especifica las filas que se van a recuperar. Como no hay ninguna cláusula **WHERE**, se recuperan todas las filas por defecto.
- La cláusula **GROUP BY** especifica cómo se deben agrupar las filas. Las filas se agrupan por número de departamento, por lo que la función **AVG** que se está aplicando a la columna de salario calculará el *salario medio de cada departamento*.

Uso de la Cláusula GROUP BY

La columna GROUP BY no tiene que estar en la lista SELECT.

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

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

ORACLE

5-16

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

La Cláusula GROUP BY (continuación)

La columna GROUP BY no tiene que estar en la cláusula SELECT. Por ejemplo, la sentencia SELECT de la transparencia muestra los salarios medios para cada departamento sin mostrar los números de los departamentos respectivos. Sin embargo, sin los números de departamento, los resultados no parecen significativos.

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

DEPARTMENT_ID	AVG(SALARY)
50	3500
10	4400
60	6400
...	
90	19333.3333

8 rows selected.

Agrupación por más de una Columna

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.

**“Sume los
salarios de
la tabla
EMPLOYEES
para cada cargo,
agrupado por
departamento.**

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.

ORACLE

Grupos dentro de Grupos

A veces necesita ver resultados para grupos dentro de grupos. En la transparencia se muestra un informe con el salario total que se paga a cada cargo, dentro de cada departamento.

La tabla EMPLOYEES se agrupa en primer lugar por número de departamento y, dentro de dicha agrupación, por cargo. Por ejemplo, los cuatro agentes de bolsa del departamento 50 están agrupados y se produce un resultado único (salario total) para todos los agentes de bolsa dentro del grupo.

Uso de la Cláusula GROUP BY en Varias Columnas

```
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.

ORACLE

5-18

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Grupos dentro de Grupos (continuación)

Puede devolver resultados resumidos para grupos y subgrupos incluyendo en una lista más de una columna GROUP BY. Puede determinar el orden por defecto de los resultados por el orden de las columnas en la cláusula GROUP BY. A continuación, se muestra el modo en que se evalúa la sentencia SELECT de la transparencia, que contiene una cláusula GROUP BY:

- La cláusula SELECT especifica la columna que se va a recuperar:
 - Número de departamento de la tabla EMPLOYEES
 - Identificador de cargo de la tabla EMPLOYEES
 - La suma de todos los salarios en el grupo que ha especificado en la cláusula GROUP BY
- La cláusula FROM especifica las tablas a las que debe acceder la base de datos: la tabla EMPLOYEES.
- La cláusula GROUP BY especifica cómo debe agrupar las filas:
 - En primer lugar, las filas se agrupan por número de departamento.
 - En segundo lugar, dentro de los grupos de números de departamento, las filas se agrupan por identificador de cargo.

De esta forma, la función SUM se está aplicando a la columna de salario para todos los identificadores de cargo dentro de cada grupo de números de departamento.

Consultas no Válidas Utilizando Funciones de Grupo

Toda columna o expresión de la lista **SELECT** que no sea una función agregada debe estar en la cláusula **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
```

La columna falta en la cláusula **GROUP BY**

ORACLE

5-19

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Consultas no Válidas Utilizando Funciones de Grupo

Siempre que utilice una mezcla de elementos individuales (**DEPARTMENT_ID**) y funciones de grupo (**COUNT**) en la misma sentencia **SELECT**, debe incluir una cláusula **GROUP BY** que especifique los elementos individuales (en este caso, **DEPARTMENT_ID**). Si falta la cláusula **GROUP BY**, aparecerá el mensaje de error “not a single-group group function” y un asterisco (*) señalará la columna incorrecta. Puede corregir el error de la transparencia agregando la cláusula **GROUP BY**.

```
SELECT  department_id, count(last_name)
FROM    employees
GROUP BY department_id;
```

DEPARTMENT_ID	COUNT(LAST_NAME)
10	1
20	2
...	
	1

8 rows selected.

Toda columna o expresión de la lista **SELECT** que no sea una función agregada deberá estar en la cláusula **GROUP BY**.

Consultas no Válidas Utilizando Funciones de Grupo

- No se puede utilizar la cláusula **WHERE** para restringir grupos.
- Utilice la cláusula **HAVING** para restringir grupos.
- No se pueden utilizar funciones de grupo en la cláusula **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
```

No se puede utilizar la cláusula **WHERE para restringir grupos**

ORACLE

5-20

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Consultas no Válidas Utilizando Funciones de Grupo (continuación)

La cláusula **WHERE** no se puede utilizar para restringir grupos. La sentencia **SELECT** de la transparencia da como resultado un error, pues utiliza la cláusula **WHERE** para restringir la visualización de salarios medios de los departamentos que tienen un salario medio mayor que \$8.000.

Puede corregir el error de la transparencia utilizando la cláusula **HAVING** para restringir grupos.

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

DEPARTMENT_ID	AVG(SALARY)
20	9500
80	10033.3333
90	19333.3333
110	10150

Exclusión de Resultados de Grupo

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.

**El salario
máximo
por departamento
cuando es
mayor que
\$10.000**

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

ORACLE

Restricción de Resultados de Grupo

De la misma forma que utiliza la cláusula `WHERE` para restringir las filas que selecciona, utilice la cláusula `HAVING` para restringir grupos. Para buscar el salario máximo de cada departamento, pero mostrando solamente los departamentos que tengan un salario máximo de más de \$10.000, debe:

1. Buscar el salario medio para cada departamento agrupando por número de departamento.
2. Restringir los grupos a los departamentos con un salario máximo mayor que \$10.000.

Exclusión de Resultados de Grupo: La Cláusula HAVING

Utilice la cláusula HAVING para restringir grupos:

1. Las filas se agrupan.
2. Se aplica la función de grupo.
3. Se muestran los grupos que coinciden con la cláusula HAVING.

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

ORACLE

5-22

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

La Cláusula HAVING

Utilice la cláusula HAVING para especificar qué grupos se van a mostrar y, de esta forma, restringirá aún más los grupos sobre la base de información agregada.

En la sintaxis:

group_condition restringe los grupos de filas devueltas a los grupos para los que es verdadera la condición especificada.

Oracle Server realiza los siguientes pasos cuando se utiliza la cláusula HAVING:

1. Las filas se agrupan.
2. Se aplica al grupo la función de grupo.
3. Se muestran los grupos que coinciden con los criterios de la cláusula HAVING.

La cláusula HAVING puede preceder a la cláusula GROUP BY, pero se recomienda que coloque en primer lugar la cláusula GROUP BY porque es más lógico. Se forman grupos y se calculan las funciones de grupo antes de que se aplique la cláusula HAVING a los grupos de la lista SELECT.

Uso de la Cláusula HAVING

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

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

ORACLE

5-23

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

La Cláusula HAVING (continuación)

En el ejemplo de la transparencia se muestran números de departamento y salarios máximos para los departamentos cuyos salarios máximos son mayores que \$10.000.

Puede utilizar la cláusula GROUP BY sin utilizar una función de grupo en la lista SELECT.

Si restringe las filas basándose en el resultado de una función de grupo, debe tener una cláusula GROUP BY además de la cláusula HAVING.

En el siguiente ejemplo se muestran los números de departamento y los salarios medios para los departamentos cuyos salarios máximos sean mayores que \$10.000:

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

DEPARTMENT_ID	AVG(SALARY)
20	9500
80	10033.3333
90	19333.3333
110	10150

Uso de la Cláusula 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

ORACLE

La Cláusula HAVING (continuación)

En el ejemplo de la transparencia se muestra el identificador de cargo y el salario mensual total para cada cargo con una nómina total superior a \$13.000. El ejemplo excluye a representantes de ventas y ordena la lista según el salario mensual total.

Anidamiento de Funciones de Grupo

Visualice el salario medio máximo.

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

MAX(AVG(SALARY))
19333.3333

ORACLE

Anidamiento de Funciones de Grupo

Las funciones de grupo se pueden anidar hasta una profundidad de dos. En el ejemplo de la transparencia se muestra el salario medio máximo.

Resumen

En esta lección, debería haber aprendido a:

- Utilizar las funciones de grupo COUNT, MAX, MIN, AVG
- Escribir consultas que utilicen la cláusula GROUP BY
- Escribir consultas que utilicen la cláusula HAVING

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

ORACLE

5-26

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Resumen

Hay siete funciones de grupo disponibles en SQL:

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

Puede crear subgrupos utilizando la cláusula GROUP BY. Los grupos se pueden excluir utilizando la cláusula HAVING.

Coloque las cláusulas HAVING y GROUP BY después de la cláusula WHERE en una sentencia y la cláusula ORDER BY al final.

Oracle Server evalúa las cláusulas en el siguiente orden:

1. Si la sentencia contiene una cláusula WHERE, el servidor establece las filas candidatas.
2. El servidor identifica los grupos especificados en la cláusula GROUP BY.
3. La cláusula HAVING restringe aún más los grupos de resultados que no cumplen los criterios de grupo de la cláusula HAVING.

Visión General de la Práctica 5

Esta práctica cubre los siguientes temas:

- Escritura de consultas que utilizan las funciones de grupo
- Agrupación por filas para obtener más de un resultado
- Exclusión de grupos utilizando la cláusula `HAVING`

ORACLE

5-27

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Visión General de la Práctica 5

Al final de esta práctica, debería estar familiarizado con el uso de funciones de grupo y con la selección de grupos de datos.

Preguntas para Contestar en Papel

Para las preguntas 1-3, marque con un círculo Verdadero o Falso.

Nota: Se utilizan alias de columna para las consultas.

Práctica 5

Determine la validez de las tres siguientes afirmaciones. Marque con un círculo Verdadero o Falso.

1. Las funciones de grupo pasan por muchas filas para producir un resultado por grupo.
Verdadero/Falso
2. Las funciones de grupo incluyen nulos en los cálculos.
Verdadero/Falso
3. La cláusula WHERE restringe las filas antes de su inclusión en un cálculo de grupo.
Verdadero/Falso
4. Visualice el salario mayor, el menor, la suma y el salario medio de todos los empleados. Etiquete las columnas Maximum, Minimum, Sum y Average, respectivamente. Redondee los resultados hacia el número entero más próximo. Coloque la sentencia SQL en un archivo de texto llamado lab5_4.sql.

Maximum	Minimum	Sum	Average
24000	2500	175500	8775

5. Modifique la consulta de lab5_4.sql para visualizar el salario mínimo, el máximo, la suma y el salario medio para cada tipo de cargo. Vuelva a guardar lab5_4.sql como lab5_5.sql. Ejecute la sentencia de lab5_5.sql.

JOB_ID	Maximum	Minimum	Sum	Average
AC_ACCOUNT	8300	8300	8300	8300
AC_MGR	12000	12000	12000	12000
AD_ASST	4400	4400	4400	4400
AD PRES	24000	24000	24000	24000
AD_VP	17000	17000	34000	17000
IT_PROG	9000	4200	19200	6400
MK_MAN	13000	13000	13000	13000
MK_REP	6000	6000	6000	6000
SA_MAN	10500	10500	10500	10500
SA_REP	11000	7000	26600	8867
ST_CLERK	3500	2500	11700	2925
ST_MAN	5800	5800	5800	5800

12 rows selected.

Práctica 5 (continuación)

6. Escriba una consulta para visualizar el número de personas con el mismo cargo.

JOB_ID	COUNT(*)
AC_ACCOUNT	1
AC_MGR	1
AD_ASST	1
AD_PRES	1
AD_VP	2
IT_PROG	3
MK_MAN	1
MK_REP	1
SA_MAN	1
SA_REP	3
ST_CLERK	4
ST_MAN	1

12 rows selected.

7. Determine el número de directores sin enumerarlos. Etiquete la columna como Number of Managers. *Indicación: Utilice la columna MANAGER_ID para determinar el número de directores.*

Number of Managers
8

8. Escriba una consulta para visualizar la diferencia entre el salario mayor y el menor. Etiquete la columna DIFFERENCE.

DIFFERENCE
21500

Si tiene tiempo, complete los siguientes ejercicios:

9. Visualice el número de director y el salario del empleado de menor sueldo para dicho director. Excluya a todas las personas con director desconocido. Excluya los grupos donde el salario mínimo sea \$6.000 o inferior. Ordene el resultado en orden descendente de salario.

MANAGER_ID	MIN(SALARY)
102	9000
205	8300
149	7000

Práctica 5 (continuación)

10. Escriba una consulta para visualizar el nombre, la ubicación, el número de empleados y el salario medio de todos los empleados de cada departamento. Etiquete las columnas como Name, Location, Number of People y Salary, respectivamente. Redondee el salario medio en dos posiciones decimales.

Name	Location	Number of People	Salary
Accounting	1700	2	10150
Administration	1700	1	4400
Executive	1700	3	19333.33
IT	1400	3	6400
Marketing	1800	2	9500
Sales	2500	3	10033.33
Shipping	1500	5	3500

7 rows selected.

Si desea una comprobación adicional, complete los siguientes ejercicios:

11. Cree una consulta que muestre el número total de empleados y, de ese total, el número de empleados contratados en 1995, 1996, 1997 y 1998. Cree las cabeceras de columna adecuadas.

TOTAL	1995	1996	1997	1998
20	1	2	2	3

12. Cree una consulta matriz para visualizar el cargo, el salario para dicho cargo basado en el número de departamento y el salario total para dicho cargo, para los departamentos 20, 50, 80 y 90, asignando a cada columna la cabecera apropiada.

Job	Dept 20	Dept 50	Dept 80	Dept 90	Total
AC_ACCOUNT					8300
AC_MGR					12000
AD_ASST					4400
AD_PRES				24000	24000
AD_VP				34000	34000
IT_PROG					19200
MK_MAN	13000				13000
MK_REP	6000				6000
SA_MAN			10500		10500
SA_REP			19600		26600
ST_CLERK		11700			11700
ST_MAN		5800			5800

12 rows selected.

6

Subconsultas

ORACLE®

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Objetivos

Al finalizar esta lección, debería estar capacitado para:

- **Describir los tipos de problemas que pueden resolver las subconsultas**
- **Definir subconsultas**
- **Listar los tipos de subconsultas**
- **Escribir subconsultas de una sola fila y de varias filas**

ORACLE

6-2

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Objetivo de la Lección

En esta lección, aprenderá sobre funciones más avanzadas de la sentencia `SELECT`. Podrá escribir subconsultas en la cláusula `WHERE` de otra sentencia `SQL` para obtener valores basados en un valor condicional desconocido. Esta lección abarca subconsultas de una sola fila y subconsultas de varias filas.

Uso de una Subconsulta para Resolver un Problema

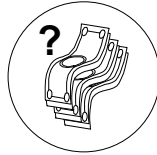
¿Quién tiene un salario mayor que el de Abel?

Consulta Principal:



¿Qué empleados tienen salarios mayores que el de Abel?

Subconsulta



¿Cuál es el salario de Abel?



ORACLE

Uso de una Subconsulta para Resolver un Problema

Suponga que desea escribir una consulta para averiguar quién gana un salario mayor que el de Abel.

Para resolver este problema, necesita *dos* consultas: una para buscar lo que gana Abel y una segunda para buscar quién gana más de esta cantidad.

Puede resolver este problema combinando las dos consultas, colocando una consulta *dentro* de la otra.

La consulta interna o *subconsulta* devuelve un valor que utiliza la consulta externa o principal. La utilización de una subconsulta es equivalente a realizar dos consultas secuenciales y usar el resultado de la primera como valor de búsqueda en la segunda.

Sintaxis de Subconsulta

```
SELECT  select_list
FROM    table
WHERE   expr operator
        (SELECT      select_list
         FROM         table);
```

- La subconsulta (consulta interna) se ejecuta una vez antes de la consulta principal.
- El resultado de la subconsulta lo utiliza la consulta principal (consulta externa).

ORACLE

6-4

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Subconsultas

Las subconsultas son sentencias `SELECT` embebidas en cláusulas de otras sentencias `SELECT`. Con ellas puede crear sentencias potentes a partir de sentencias simples. Esto puede resultar muy útil si necesita seleccionar filas de una tabla con una condición que depende de los datos de la propia tabla.

Puede colocar la subconsulta en diversas cláusulas SQL como, por ejemplo:

- La cláusula `WHERE`
- La cláusula `HAVING`
- La cláusula `FROM`

En la sintaxis:


operator incluye una condición de comparación como `>`, `=` o `IN`

Nota: Las condiciones de comparación son de dos clases: operadores de una sola fila (`>`, `=`, `>=`, `<`, `<>`, `<=`) y operadores de varias filas (`IN`, `ANY`, `ALL`).

Con frecuencia se hace referencia a la subconsulta como sentencia `SELECT` anidada, sub-`SELECT` o `SELECT` interna. La subconsulta se suele ejecutar primero y su resultado se utiliza para completar la condición de consulta de la consulta principal o externa.

Uso de una Subconsulta

```
SELECT last_name
FROM   employees 11000
WHERE  salary >
      (SELECT salary
       FROM   employees
       WHERE  last_name = 'Abel');
```



LAST_NAME
King
Kochhar
De Haan
Hartstein
Higgins

ORACLE

Uso de una Subconsulta

En la transparencia, la consulta interna determina el salario del empleado Abel. La consulta externa toma el resultado de la consulta interna y lo utiliza para mostrar todos los empleados que ganan más de esta cantidad.

Instrucciones para Utilizar Subconsultas

- **Escriba las subconsultas entre paréntesis.**
- **Coloque las subconsultas a la derecha de la condición de comparación.**
- **La cláusula `ORDER BY` de la subconsulta no es necesaria salvo que esté realizando un análisis N principal.**
- **Utilice operadores de una sola fila con subconsultas de una sola fila y operadores de varias filas con subconsultas de varias filas.**

ORACLE

6-6

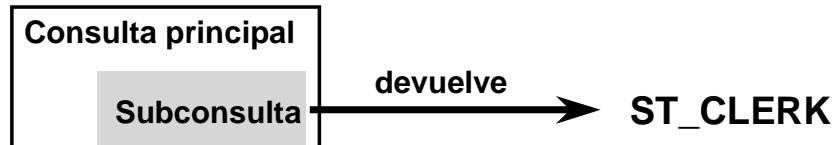
Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Instrucciones para Utilizar Subconsultas

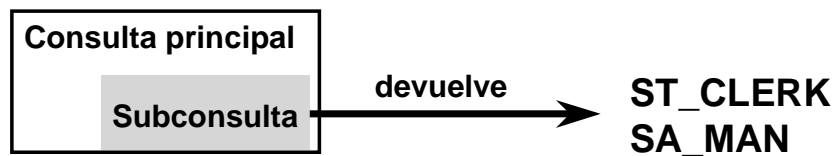
- La subconsulta se debe escribir entre paréntesis.
- Coloque la subconsulta a la derecha de la condición de comparación para una mejor legibilidad.
- Antes de la versión Oracle8i, las subconsultas no podían contener una cláusula `ORDER BY`. Sólo se puede utilizar una cláusula `ORDER BY` para una sentencia `SELECT` y, si se especifica, debe ser la última cláusula de la sentencia `SELECT` principal. A partir de la versión Oracle8i, se puede utilizar una cláusula `ORDER BY` y es necesaria en la subconsulta para realizar un análisis N principal.
- En las subconsultas se utilizan dos clases de condiciones de comparación: operadores de una sola fila y operadores de varias filas.

Tipos de Subconsultas

- Subconsulta de una sola fila



- Subconsulta de varias filas



ORACLE

Tipos de Subconsultas

- Subconsultas de una sola fila: Consultas que devuelven una sola fila a partir de la sentencia `SELECT` interna.
- Subconsultas de varias filas: Consultas que devuelven más de una fila a partir de la sentencia `SELECT` interna.

Nota: También hay subconsultas de varias columnas: Consultas que devuelven más de una columna a partir de la sentencia `SELECT` interna.

Subconsultas de una Sola Fila

- Devuelven una sola fila
- Utilizan operadores de comparación de una sola fila

Operador	Significado
=	Igual que
>	Mayor que
>=	Mayor o igual que
<	Menor que
<=	Menor o igual que
<>	No igual a

ORACLE

6-8

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Subconsultas de una Sola Fila

Una subconsulta de una sola fila es aquella que devuelve una fila a partir de la sentencia `SELECT` interna. Este tipo de subconsulta utiliza un operador de una sola fila. La transparencia proporciona una lista de operadores de una sola fila.

Ejemplo

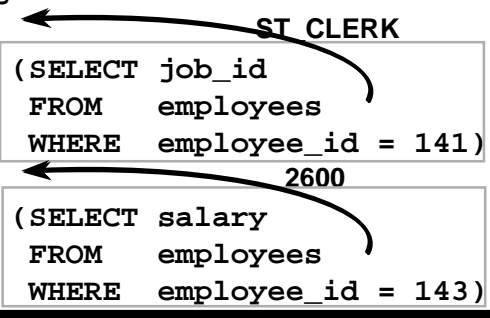
Muestre los empleados cuyos identificadores de cargo sean los mismos que el del empleado 141.

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
      (SELECT job_id
       FROM   employees
       WHERE  employee_id = 141);
```

LAST_NAME	JOB_ID
Rajs	ST_CLERK
Davies	ST_CLERK
Matos	ST_CLERK
Vargas	ST_CLERK

Ejecución de Subconsultas de una Sola Fila

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id = (SELECT job_id
                 FROM   employees
                 WHERE  employee_id = 141)
AND    salary > (SELECT salary
                 FROM   employees
                 WHERE  employee_id = 143);
```



LAST_NAME	JOB_ID	SALARY
Rajs	ST_CLERK	3500
Davies	ST_CLERK	3100

ORACLE

Ejecución de Subconsultas de una Sola Fila

Se puede considerar una consulta `SELECT` como un bloque de consultas. El ejemplo de la transparencia muestra empleados cuyos identificadores de cargo son los mismos que el del empleado 141 y cuyos salarios son mayores que el del empleado 143.


El ejemplo consta de tres bloques de consultas: la consulta externa y dos consultas internas. Los bloques de la consulta interna se ejecutan en primer lugar y producen los resultados de consulta `ST_CLERK` y `2600`, respectivamente. A continuación, se procesa el bloque de la consulta externa y éste utiliza los valores devueltos por las consultas internas para completar sus condiciones de búsqueda.

Las dos consultas internas devuelven valores únicos (`ST_CLERK` y `2600`, respectivamente), por lo que esta sentencia SQL se llama subconsulta de una sola fila.

Nota: Las consultas externas e internas pueden obtener datos de tablas diferentes.

Uso de Funciones de Grupo en una Subconsulta

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary = (SELECT MIN(salary)
                  FROM   employees);
```

A curved arrow points from the value '2500' to the 'salary =' comparison in the WHERE clause of the main query.

LAST_NAME	JOB_ID	SALARY
Vargas	ST_CLERK	2500

ORACLE

Uso de Funciones de Grupo en una Subconsulta

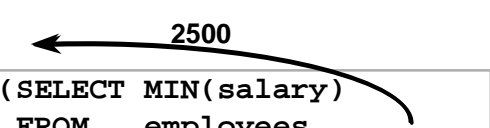
Puede mostrar datos de una consulta principal utilizando una función de grupo en una subconsulta para devolver una sola fila. La subconsulta está entre paréntesis y colocada detrás de la condición de comparación.

El ejemplo de la transparencia muestra el apellido del empleado, el identificador de cargo y el salario de todos los empleados cuyos salarios sean iguales al mínimo. La función de grupo MIN devuelve un valor único (2500) a la consulta externa.

La Cláusula HAVING con Subconsultas

- Oracle Server ejecuta en primer lugar las subconsultas.
- Oracle Server devuelve resultados a la cláusula HAVING de la consulta principal.

```
SELECT  department_id, MIN(salary)
FROM    employees
GROUP BY department_id
HAVING  MIN(salary) > (SELECT MIN(salary)
                       FROM    employees
                       WHERE   department_id = 50);
```



ORACLE

6-11

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

La Cláusula HAVING con Subconsultas

Puede utilizar subconsultas no sólo en la cláusula WHERE, sino también en la cláusula HAVING. Oracle Server ejecuta la subconsulta y los resultados se devuelven a la cláusula HAVING de la consulta principal.

La sentencia SQL de la transparencia muestra todos los departamentos que tienen un salario mínimo mayor que el del departamento 50.

DEPARTMENT_ID	MIN(SALARY)
10	4400
20	6000
...	
	7000

7 rows selected.

Ejemplo

Busque el cargo con el salario medio más bajo.

```
SELECT  job_id, AVG(salary)
FROM    employees
GROUP BY job_id
HAVING  AVG(salary) = (SELECT  MIN(AVG(salary))
                       FROM    employees
                       GROUP BY job_id);
```

¿Qué Es Incorrecto en esta Sentencia?

```
SELECT employee_id, last_name
FROM employees
WHERE salary =
    (SELECT MIN(salary)
     FROM employees
     GROUP BY department_id);
```

```
ERROR at line 4:
ORA-01427: single-row subquery returns more than
one row
```

Operador de una sola fila con subconsulta de varias filas

ORACLE

6-12

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Errores de Subconsultas

Un error común de las subconsultas es que se devuelve más de una fila para una subconsulta de una sola fila.

En la sentencia SQL de la transparencia, la subconsulta contiene una cláusula GROUP BY, que implica que devolverá varias filas, una por cada grupo que busque. En este caso, el resultado de la subconsulta será 4400, 6000, 2500, 4200, 7000, 17000 y 8300.

La consulta externa toma los resultados de la subconsulta (4400, 6000, 2500, 4200, 7000, 17000, 8300) y los utiliza en su cláusula WHERE. Esta cláusula contiene un operador igual (=), un operador de comparación de una sola fila que espera un solo valor. El operador = no puede aceptar más de un valor de la subconsulta y, por lo tanto, genera el error.

Para corregir este error, cambie el operador = por IN.

¿Devuelve Filas esta Sentencia?

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
      (SELECT job_id
       FROM   employees
       WHERE  last_name = 'Haas');
```

```
no rows selected
```

La subconsulta no devuelve ningún valor

ORACLE

6-13

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Problemas de Subconsultas

Un problema común de las subconsultas es que la consulta interna no devuelve ninguna fila.

En la sentencia SQL de la transparencia, la subconsulta contiene una cláusula WHERE.

Probablemente, la intención es buscar el empleado de nombre Haas. La sentencia es correcta, pero no selecciona ninguna fila cuando se ejecuta.

No hay ningún empleado llamado Haas, por lo que la subconsulta no devuelve ninguna fila. La consulta externa toma los resultados de la subconsulta (valor nulo) y los utiliza en su cláusula WHERE. La consulta externa no encuentra ningún empleado con el identificador de cargo igual a valor nulo y no devuelve ninguna fila. Si existiera un cargo con el valor nulo, la fila no se devolvería porque la comparación de dos valores nulos origina un valor nulo, por lo tanto la condición WHERE no es verdadera.

Subconsultas de Varias Filas

- Devuelven más de una fila
- Utilizan operadores de comparación de varias filas

Operador	Significado
IN	Igual a cualquier miembro de la lista
ANY	Compara el valor con cada valor devuelto por la subconsulta
ALL	Compara el valor con todos los valores devueltos por la subconsulta

ORACLE

6-14

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Subconsultas de Varias Filas

Las subconsultas que devuelven más de una fila se denominan subconsultas de varias filas. Se utiliza un operador de varias filas, en lugar de uno de una sola fila, en una subconsulta de varias filas. El operador de varias filas espera un valor o varios.

```
SELECT last_name, salary, department_id
FROM employees
WHERE salary IN (SELECT MIN(salary)
                  FROM employees
                  GROUP BY department_id);
```

Ejemplo

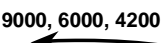
Busque los empleados que ganan el mismo salario que el mínimo de cada departamento.

La consulta interna se ejecuta en primer lugar y proporciona un resultado de consulta. A continuación, se procesa el bloque de consulta principal y utiliza los valores devueltos por la consulta interna para completar su condición de búsqueda. En realidad, la consulta principal aparecería en Oracle Server como se indica a continuación:

```
SELECT last_name, salary, department_id
FROM employees
WHERE salary IN (2500, 4200, 4400, 6000, 7000, 8300, 8600, 17000);
```


Uso del Operador ANY en Subconsultas de Varias Filas

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ANY
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```



EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
124	Mourgos	ST_MAN	5800
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

10 rows selected.

ORACLE

6-15

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Subconsultas de Varias Filas (continuación)

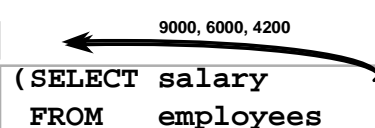
El operador ANY (y su sinónimo, el operador SOME) compara un valor con *cada* valor devuelto por una subconsulta. El ejemplo de la transparencia muestra los empleados que no son programadores de IT y cuyos salarios son menores que los de cualquier programador de IT. El salario máximo que gana un programador es de \$9.000.

<ANY significa menos que el máximo. >ANY significa más que el mínimo. =ANY es equivalente a IN.

<ALL significa menos que el máximo. >ALL significa más que el mínimo.

Uso del Operador ALL en Subconsultas de Varias Filas

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ALL
      (SELECT salary
       FROM   employees
       WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```



EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
141	Rajs	ST_CLERK	3500
142	Davies	ST_CLERK	3100
143	Matos	ST_CLERK	2600
144	Vargas	ST_CLERK	2500

ORACLE

Subconsultas de Varias Filas (continuación)

El operador ALL compara un valor con *todos* los valores devueltos por una subconsulta. El ejemplo de la transparencia muestra los empleados cuyos salarios son menores que el salario de todos los empleados con el identificador de cargo IT_PROG y cuyos cargos no son IT_PROG.

>ALL significa más que el máximo y <ALL significa menos que el mínimo.

El operador NOT se puede utilizar con los operadores IN, ANY y ALL.

Valores Nulos en una Subconsulta

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id NOT IN
                                (SELECT mgr.manager_id
                                FROM   employees mgr);

no rows selected
```

ORACLE

6-17

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Devolución de Valores Nulos en el Juego Resultante de una Subconsulta

La sentencia SQL de la transparencia intenta mostrar todos los empleados que no tienen subordinados. Lógicamente, esta sentencia SQL debería haber devuelto 12 filas. Sin embargo, no devuelve ninguna. Uno de los valores devueltos por la consulta interna es un valor nulo, por lo que la consulta entera no devuelve ninguna fila. El motivo es que todas las condiciones que comparen un valor nulo producen un valor nulo. Por ello, si es posible que en el juego de resultados de una subconsulta haya valores nulos, no utilice el operador NOT IN. Este operador es equivalente a <> ALL.

Observe que el valor nulo como parte del juego de resultados de una subconsulta no supone un problema si utiliza el operador IN. Este operador es equivalente a =ANY. Por ejemplo, para mostrar los empleados que tienen subordinados, utilice la siguiente sentencia SQL:

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id IN
                                (SELECT mgr.manager_id
                                FROM   employees mgr);
```

Igualmente, se puede incluir una cláusula WHERE en la subconsulta para mostrar todos los empleados que no tienen subordinados:

```
SELECT last_name FROM employees
WHERE  employee_id NOT IN
                                (SELECT manager_id
                                FROM   employees
                                WHERE  manager_id IS NOT NULL);
```

Introducción a Oracle9i: SQL 6-17

Resumen

En esta lección, debería haber aprendido a:

- Identificar cuándo una subconsulta puede ayudar a resolver una pregunta
- Escribir subconsultas cuando una consulta está basada en valores desconocidos

```
SELECT  select_list
FROM    table
WHERE   expr operator
        (SELECT select_list
         FROM   table);
```

ORACLE

Resumen

En esta lección, debería haber aprendido a utilizar las subconsultas. Una subconsulta es una sentencia `SELECT` embebida en una cláusula de otra sentencia `SQL`. Las subconsultas resultan útiles cuando una consulta está basada en un criterio de búsqueda con valores intermedios desconocidos.

Las subconsultas tienen las siguientes características:

- Pueden transferir una fila de datos a una sentencia principal que contenga un operador de una sola fila, como `=`, `<>`, `>`, `>=`, `<` o `<=`.
- Pueden transferir varias filas de datos a una sentencia principal que contenga un operador de varias filas como, por ejemplo, `IN`.
- Oracle Server las procesa en primer lugar y la cláusula `WHERE` o `HAVING` utiliza los resultados.
- Pueden contener funciones de grupo.

Visión General de la Práctica 6

Esta práctica cubre los siguientes temas:

- **Creación de subconsultas para consultar valores basados en criterios desconocidos.**
- **Uso de subconsultas para averiguar qué valores existen en un juego de datos y no en otro.**

ORACLE

6-19

Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Práctica 6

En esta práctica, escribirá consultas complejas utilizando sentencias anidadas SELECT.

Preguntas para Contestar en Papel

Puede crear primero la consulta interna para estas preguntas. Asegúrese de que se ejecuta y produce los datos que espera antes de codificar la consulta externa.

Práctica 6

1. Escriba una consulta que muestre el apellido y la fecha de contratación de cualquier empleado del mismo departamento que Zlotkey. Excluya a Zlotkey.

LAST_NAME	HIRE_DATE
Abel	11-MAY-96
Taylor	24-MAR-98

2. Cree una consulta para mostrar los números de empleado y los apellidos de todos los empleados que ganen más del salario medio. Ordene los resultados por salario en orden ascendente.

EMPLOYEE_ID	LAST_NAME	SALARY
103	Hunold	9000
149	Zlotkey	10500
174	Abel	11000
205	Higgins	12000
201	Hartstein	13000
101	Kochhar	17000
102	De Haan	17000
100	King	24000

8 rows selected.

3. Escriba una consulta que muestre los números de empleado y los apellidos de todos los empleados que trabajen en un departamento con cualquier empleado cuyo apellido contenga una *u*. Coloque la sentencia SQL en un archivo de texto llamado lab6_3.sql. Ejecute la consulta.

EMPLOYEE_ID	LAST_NAME
124	Mourgos
141	Rajs
142	Davies
143	Matos
144	Vargas
103	Hunold
104	Ernst
107	Lorentz

8 rows selected.

Práctica 6 (continuación)

- Muestre el apellido, el número de departamento y el identificador de cargo de todos los empleados cuyos identificadores de ubicación de departamento sean 1700.

LAST_NAME	DEPARTMENT_ID	JOB_ID
Whalen	10	AD_ASST
King	90	AD PRES
Kochhar	90	AD_VP
De Haan	90	AD_VP
Higgins	110	AC_MGR
Gietz	110	AC_ACCOUNT

6 rows selected.

- Muestre el apellido y el salario de todos los empleados que informen a King.

LAST_NAME	SALARY
Kochhar	17000
De Haan	17000
Mourgos	5800
Zlotkey	10500
Hartstein	13000

- Muestre el número de departamento, el apellido y el identificador de cargo de todos los empleados del departamento Executive.

DEPARTMENT_ID	LAST_NAME	JOB_ID
90	King	AD PRES
90	Kochhar	AD_VP
90	De Haan	AD_VP

Si tiene tiempo, complete los siguientes ejercicios:

- Modifique la consulta en lab6_3.sql para mostrar los números de empleado, los apellidos y los salarios de todos los empleados que ganan más del salario medio y que trabajan en un departamento con un empleado que tenga una *u* en su apellido. Vuelva a guardar lab6_3.sql como lab6_7.sql. Ejecute la sentencia en lab6_7.sql.

EMPLOYEE_ID	LAST_NAME	SALARY
103	Hunold	9000

