UNIDAD 4: VISUALIZACION DE 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.

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

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.

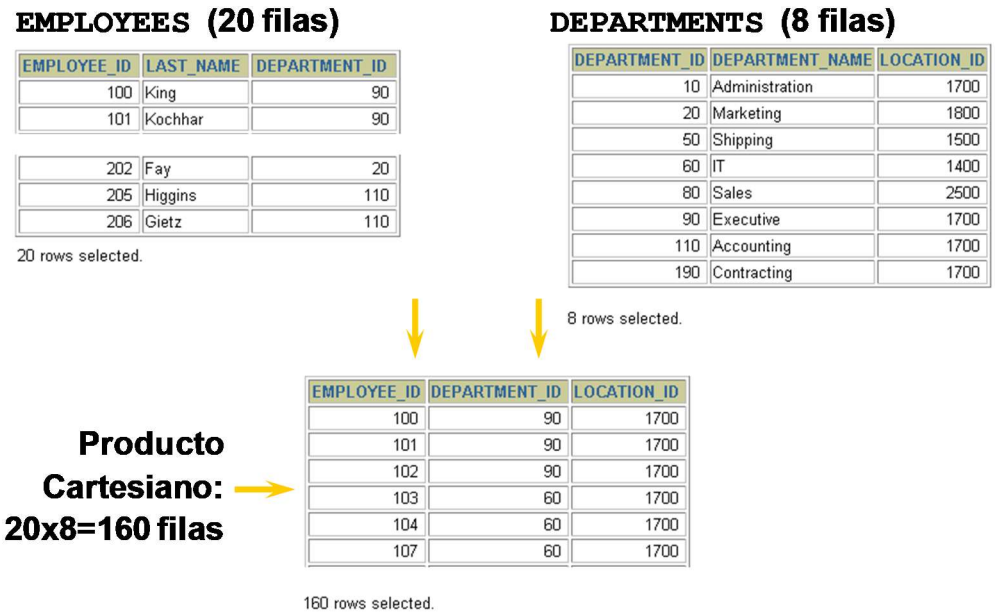
Los productos Cartesianos son útiles para algunas pruebas en las que se necesita generar un gran número de filas para simular una cantidad razonable de datos.

GENERACION DE UN PRODUCTO CARTESIANO

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;
```

Generación de un Producto Cartesiano



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.

Uniones de Propiedad de Oracle (8*i* 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

UNION DE TABLAS UTILIZANDO LA SINTAXIS ORACLE

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

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".

UNION 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*.

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

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping
60	IT
60	IT
60	IT
80	Sales
80	Sales
80	Sales

RECUPERACION DE REGISTROS CON UNIONES DE IGUALDAD

En el ejemplo de la figura siguiente:

- 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.

```
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.

CONDICIONES DE BUSQUEDA ADICIONALES UTILIZANDO EL OPERADOR AND

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';
```

EMPLOYEES

LAST_NAME	DEPARTMENT_ID
Whalen	10
Hartstein	20
Fay	20
Mourgos	50
Rajs	50
Davies	50
Matos	50
Vargas	50
Hunold	60
Ernst	60

...

DEPARTMENTS

DEPARTMENT_ID	DEPARTMENT_NAME
10	Administration
20	Marketing
20	Marketing
50	Shipping
50	Shipping
50	Shipping
50	Shipping
50	Shipping
60	IT
60	IT

...

CUALIFICACION 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

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.

UNION DE MAS DE DOS TABLAS

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;
```

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

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.**

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 (=).

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.

RECUPERACION DE REGISTROS CON UNIONES DE NO IGUALDAD

En el ejemplo de la figura, se 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.

```
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.

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, se ven 19.

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

DEPARTMENTS

DEPARTMENT_NAME	DEPARTMENT_ID
Administration	10
Marketing	20
Shipping	50
IT	60
Sales	80
Executive	90
Accounting	110
Contracting	190

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

No hay empleados en el departamento 190.

SINTAXIS DE UNIONES EXTERNAS

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:

table1.column =	es la condición que une (o relaciona) las tablas entre sí.
table2.column (+)	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.)

- **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(+);
```

USO DE UNIONES EXTERNAS

En el ejemplo de la figura 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.

```
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

AUTOUNIONES

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

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.

UNION DE UNA TABLA CONSIGO MISMA

El ejemplo de la figura 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".

```
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.

UNION DE TABLAS UTILIZANDO LA SINTAXIS SQL:1999

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

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)];
```

CREACION DE UNIONES CRUZADAS

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

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

- **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.

CREACION 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 Oracle9*i* 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

RECUPERACION DE REGISTROS CON UNIONES NATURALES

En el ejemplo de la figura, 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);
```

```
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.

CREACION DE UNIONES CON LA CLAUSULA 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.

RECUPERACION DE REGISTROS CON LA CLAUSULA USING

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;
```

```
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.

CREACION DE UNIONES CON LA CLAUSULA 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. Esta cláusula facilita la comprensión del código.

RECUPERACION DE REGISTROS CON LA CLAUSULA 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);
```

```
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.

CREACION DE UNIONES EN TRES SENTIDOS CON LA CLAUSULA ON

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;
```

```
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.

UNIONES INNER FRENTE A OUTER

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

- **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.**

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;
```

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e
LEFT OUTER JOIN departments d
ON    (e.department_id = d.department_id) ;
```

LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
Whalen	10	Administration
Fay	20	Marketing
Hartstein	20	Marketing
...		
De Haan	90	Executive
Kochhar	90	Executive
King	90	Executive
Gietz	110	Accounting
Higgins	110	Accounting
Grant		

20 rows selected.

RIGHT OUTER JOIN

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  (+);
```

```
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.

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.

```
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.

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.

```
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