

15

Uso de Operadores SET

ORACLE

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

Objetivos

Al finalizar esta lección, debería estar capacitado para:

- **Describir los operadores SET**
- **Utilizar un operador SET para combinar varias consultas en una sola**
- **Controlar el orden de las filas devueltas**

ORACLE

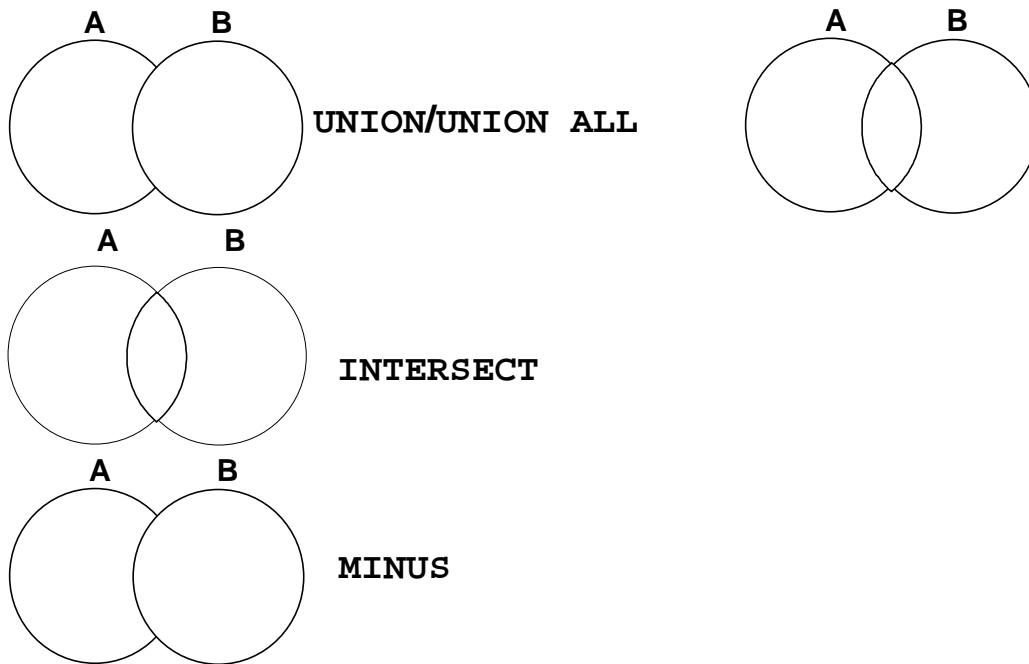
15-2

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

Objetivo de la Lección

En esta lección, aprenderá a escribir consultas mediante operadores SET.

Los Operadores SET



ORACLE

15-3

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

Los Operadores SET

Los operadores SET combinan los resultados de dos o más consultas componentes en un resultado. Las consultas que contienen operadores SET se llaman *consultas compuestas*.

Operador	Devuelve
UNION	Todas las filas distintas seleccionadas por cada una de las consultas
UNION ALL	Todas las filas seleccionadas por cualquiera de las consultas, incluidos todos los duplicados
INTERSECT	Todas las filas distintas seleccionadas por ambas consultas
MINUS	Todas las filas distintas seleccionadas por la primera sentencia SELECT y no seleccionadas por la segunda sentencia SELECT

Todos los operadores SET tienen la misma prioridad. Si una sentencia SQL contiene varios operadores SET, Oracle Server los evalúa de izquierda (arriba) a derecha (abajo) si no hay paréntesis que especifiquen explícitamente otro orden. Debe utilizar paréntesis para especificar el orden de evaluación explícitamente en columnas que utilicen el operador INTERSECT con otros operadores SET.

Nota: En la transparencia, el color claro (gris) del diagrama representa el resultado de la consulta.

Tablas Utilizadas en Esta Lección

Las tablas que se utilizan en esta lección son:

- **EMPLOYEES:** Proporciona detalles relativos a todos los empleados actuales.
- **JOB_HISTORY:** Registra los detalles de las fechas inicial y final del cargo anterior y el número de identificación de cargo y el departamento cuando un empleado cambia de cargo.

ORACLE

15-4

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

Tablas Utilizadas en Esta Lección

Se utilizan dos tablas en esta lección: EMPLOYEES y JOB_HISTORY.

La tabla EMPLOYEES almacena detalles de empleado. Para los registros de recursos humanos, esta tabla almacena un número de identificación único y una dirección de correo electrónico para cada empleado. También se almacenan detalles del empleado de número de identificación de cargo, salario y director. Algunos de los empleados perciben una comisión además del salario; también se realiza un seguimiento de esta información. La compañía organiza los roles de los empleados en cargos. Algunos empleados llevan mucho tiempo con la compañía y han ido cambiando de cargo. Esto se controla mediante la tabla JOB_HISTORY. Cuando un empleado cambia de cargo, los detalles de las fechas inicial y final del trabajo anterior, el número de identificación de cargo y el departamento se registran en la tabla JOB_HISTORY.

La estructura y los datos de las tablas EMPLOYEES y JOB_HISTORY se muestran en la página siguiente.

Ha habido casos en la compañía de gente que han ocupado el mismo cargo más de una vez durante su trabajo para la compañía. Por ejemplo piense en el empleado Taylor, que entró en la compañía el 24-MAR-1998. Taylor ocupó el cargo SA_REP del 24-MAR-98 al 31-DEC-98 y el cargo SA_MAN del 01-JAN-99 al 31-DEC-99. Taylor volvió al cargo SA_REP, que es su cargo actual.

De forma parecida, piense en el empleado Whalen, que entró en la compañía el 17-SEP-1987. Whalen ocupó el cargo AD_ASST del 17-SEP-87 al 17-JUN-93 y el cargo AC_ACCOUNT del 01-JUL-94 al 31-DEC-98. Whalen volvió al cargo AD_ASST, que es su cargo actual.

Tablas Utilizadas en Esta Lección (continuación)

DESC employees

Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
FIRST_NAME		VARCHAR2(20)
LAST_NAME	NOT NULL	VARCHAR2(25)
EMAIL	NOT NULL	VARCHAR2(25)
PHONE_NUMBER		VARCHAR2(20)
HIRE_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
SALARY		NUMBER(8,2)
COMMISSION_PCT		NUMBER(2,2)
MANAGER_ID		NUMBER(6)
DEPARTMENT_ID		NUMBER(4)
DEPARTMENT_NAME		VARCHAR2(14)

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

EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	DEPARTMENT_ID
100	King	AD_PRES	17-JUN-87	90
101	Kochhar	AD_VP	21-SEP-89	90
102	De Haan	AD_VP	13-JAN-93	90
103	Hunold	IT_PROG	03-JAN-90	60
104	Ernst	IT_PROG	21-MAY-91	60
107	Lorentz	IT_PROG	07-FEB-99	60
124	Mourgos	ST_MAN	16-NOV-99	50
141	Rajs	ST_CLERK	17-OCT-95	50
142	Davies	ST_CLERK	29-JAN-97	50
143	Matos	ST_CLERK	15-MAR-98	50
144	Vargas	ST_CLERK	09-JUL-98	50
149	Zlotkey	SA_MAN	29-JAN-00	80
174	Abel	SA_REP	11-MAY-96	80
176	Taylor	SA_REP	24-MAR-98	80
EMPLOYEE_ID	LAST_NAME	JOB_ID	HIRE_DATE	DEPARTMENT_ID
178	Grant	SA_REP	24-MAY-99	
200	Whalen	AD_ASST	17-SEP-87	10
201	Hartstein	MK_MAN	17-FEB-96	20

...

Tablas Utilizadas en Esta Lección (continuación)

DESC job_history

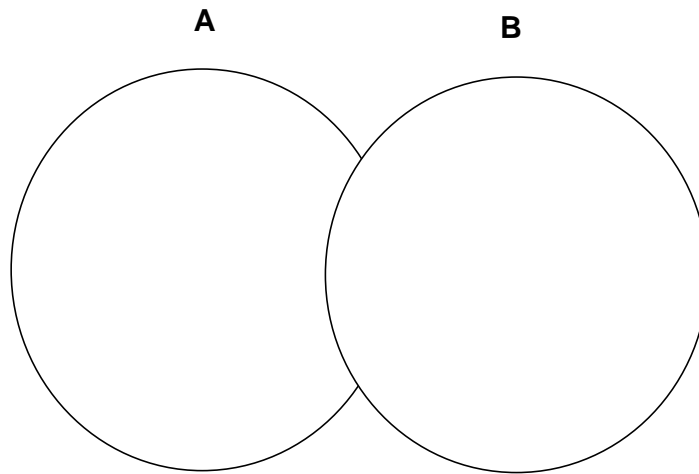
Name	Null?	Type
EMPLOYEE_ID	NOT NULL	NUMBER(6)
START_DATE	NOT NULL	DATE
END_DATE	NOT NULL	DATE
JOB_ID	NOT NULL	VARCHAR2(10)
DEPARTMENT_ID		NUMBER(4)

SELECT * FROM job_history;

EMPLOYEE_ID	START_DAT	END_DATE	JOB_ID	DEPARTMENT_ID
102	13-JAN-93	24-JUL-98	IT_PROG	60
101	21-SEP-89	27-OCT-93	AC_ACCOUNT	110
101	28-OCT-93	15-MAR-97	AC_MGR	110
201	17-FEB-96	19-DEC-99	MK_REP	20
114	24-MAR-98	31-DEC-99	ST_CLERK	50
122	01-JAN-99	31-DEC-99	ST_CLERK	50
200	17-SEP-87	17-JUN-93	AD_ASST	90
176	24-MAR-98	31-DEC-98	SA_REP	80
176	01-JAN-99	31-DEC-99	SA_MAN	80
200	01-JUL-94	31-DEC-98	AC_ACCOUNT	90

10 rows selected.

El Operador UNION



El operador UNION devuelve resultados de ambas consultas tras eliminar los duplicados.

ORACLE

15-7

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

El Operador UNION

El operador UNION devuelve todas las filas seleccionadas por cualquiera de las consultas. Utilice el operador UNION para devolver todas las filas de varias tablas y eliminar las filas duplicadas.

Instrucciones

- El número de columnas y los tipos de datos de las columnas seleccionadas deben ser idénticos en todas las sentencias SELECT utilizadas en la consulta. Los nombres de las columnas no tienen que ser idénticos.
- UNION opera en todas las columnas seleccionadas.
- Los valores NULL no se ignoran durante la comprobación de duplicados.
- El operador IN tiene una prioridad más alta que el operador UNION.
- Por defecto, el resultado se ordena en orden ascendente por la primera columna de la cláusula SELECT.

Uso del Operador UNION

Visualice detalles de los cargos actuales y anteriores de todos los empleados. Visualice cada empleado una sola vez.

```
SELECT employee_id, job_id
FROM   employees
UNION
SELECT employee_id, job_id
FROM   job_history;
```

EMPLOYEE_ID		JOB_ID
	100	AD_PRES
	101	AC_ACCOUNT
...		
	200	AC_ACCOUNT
	200	AD_ASST
...		
	205	AC_MGR
	206	AC_ACCOUNT

ORACLE

15-8

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

Uso del Operador UNION

El operador UNION elimina los registros duplicados. Si hay registros que se producen tanto en la tabla EMPLOYEES como en la tabla JOB_HISTORY y son idénticos, se mostrarán una sola vez. Observe en el resultado que se muestra que el registro para el empleado EMPLOYEE_ID 200 aparece dos veces porque su JOB_ID es diferente en cada fila.

Observe el siguiente ejemplo:

```
SELECT employee_id, job_id, department_id
FROM   employees
UNION
SELECT employee_id, job_id, department_id
FROM   job_history;
```

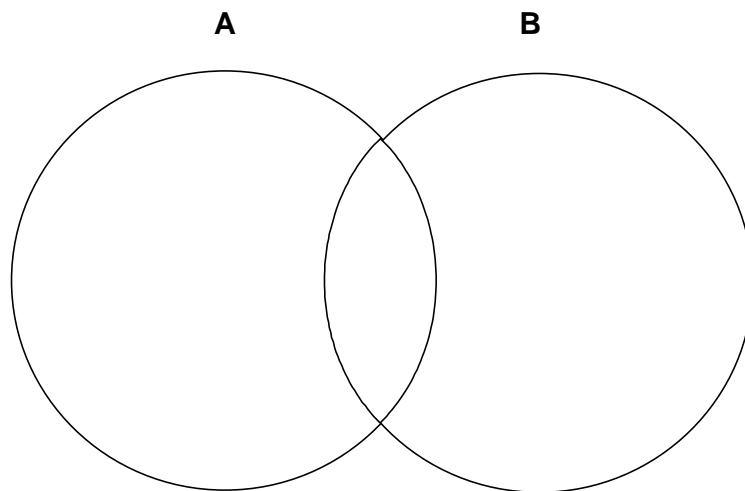
EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
...		
200	AC_ACCOUNT	90
200	AD_ASST	10
200	AD_ASST	90

29 rows selected.

Uso del Operador UNION (continuación)

En el resultado anterior, el empleado 200 aparece tres veces. ¿Por qué? Observe los valores DEPARTMENT_ID del empleado 200. Una fila tiene un DEPARTMENT_ID de 90, otra de 10 y la tercera de 90. Por estas combinaciones únicas de identificadores de cargo, cada fila es única para el empleado 200 y, por lo tanto, no se considera un duplicado. Observe que el resultado se ordena en orden ascendente por la primera columna de la cláusula SELECT, EMPLOYEE_ID en este caso.

El Operador UNION ALL



El operador UNION ALL devuelve resultados de ambas consultas, incluidos todos los duplicados.

ORACLE

15-10

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

El Operador UNION ALL

Utilice el operador UNION ALL para devolver todas las filas de varias consultas.

Instrucciones

- A diferencia de UNION, las filas duplicadas no se eliminan y el resultado no se ordena por defecto.
- No se puede utilizar la palabra clave DISTINCT.

Nota: Con la excepción de lo anteriormente expuesto, las instrucciones para UNION y UNION ALL son las mismas.

Uso del Operador UNION ALL

Visualice los departamentos actuales y anteriores de todos los empleados.

```
SELECT employee_id, job_id, department_id
FROM employees
UNION ALL
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
100	AD_PRES	90
101	AD_VP	90
...		
200	AD_ASST	10
200	AD_ASST	90
200	AC_ACCOUNT	90
...		
205	AC_MGR	110
206	AC_ACCOUNT	110

30 rows selected.

ORACLE

15-11

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

El Operador UNION ALL (continuación)

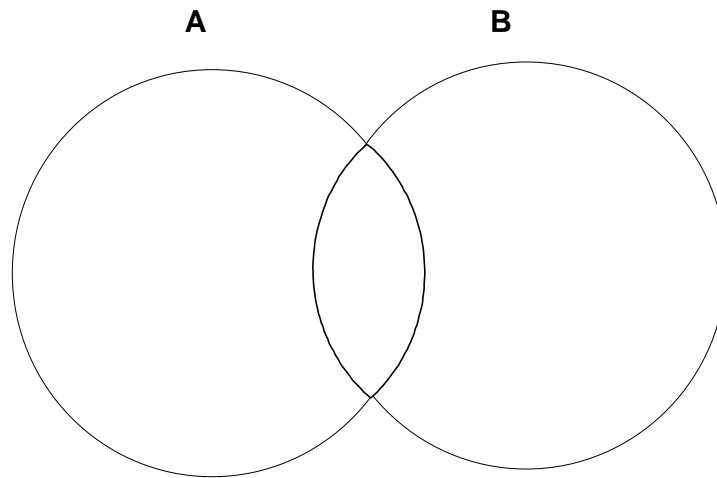
En el ejemplo, se seleccionan 30 filas. La combinación de las dos tablas ofrece un total de 30 filas. El operador UNION no elimina las filas duplicadas, que se resaltan en el resultado que se muestra en la transparencia. UNION devuelve todas las filas distintas seleccionadas por cualquiera de las consultas. UNION ALL devuelve todas las filas seleccionadas por cualquiera de las consultas, incluidos todos los duplicados. Observe la consulta de la transparencia, escrita ahora con la cláusula UNION:

```
SELECT employee_id, job_id, department_id
FROM employees
UNION
SELECT employee_id, job_id, department_id
FROM job_history
ORDER BY employee_id;
```

Esta consulta devuelve 29 filas. Esto se debe a que elimina la siguiente fila (por estar duplicada):

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
176	SA_REP	80

El Operador INTERSECT



ORACLE

15-12

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

El Operador INTERSECT

Utilice el operador INTERSECT para devolver todas las filas comunes de varias consultas.

Instrucciones

- El número de columnas y los tipos de datos de las columnas seleccionadas por las sentencias SELECT deben ser idénticos en todas las sentencias SELECT utilizadas en las consultas. Los nombres de las columnas no tienen que ser idénticos.
- Revertir el orden de las tablas intersecadas no modificará el resultado.
- INTERSECT no ignora los valores NULL.

Uso del Operador INTERSECT

Visualice los identificadores de empleado y de cargo de los empleados que actualmente tengan el cargo que ocupaban antes de comenzar a trabajar con la compañía.

```
SELECT employee_id, job_id
FROM   employees
INTERSECT
SELECT employee_id, job_id
FROM   job_history;
```

EMPLOYEE_ID	JOB_ID
176	SA_REP
200	AD_ASST

ORACLE

15-13

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

El Operador INTERSECT (continuación)

En el ejemplo de esta transparencia, la consulta devuelve sólo los registros que tengan los mismos valores en las columnas seleccionadas en ambas tablas.

¿Cuáles serán los resultados si agrega la columna DEPARTMENT_ID a la sentencia SELECT de la tabla EMPLOYEES y agrega la columna DEPARTMENT_ID a la sentencia SELECT de la tabla JOB_HISTORY y ejecuta la consulta? Los resultados pueden ser diferentes por la introducción de otra columna cuyos valores pueden ser o no duplicados.

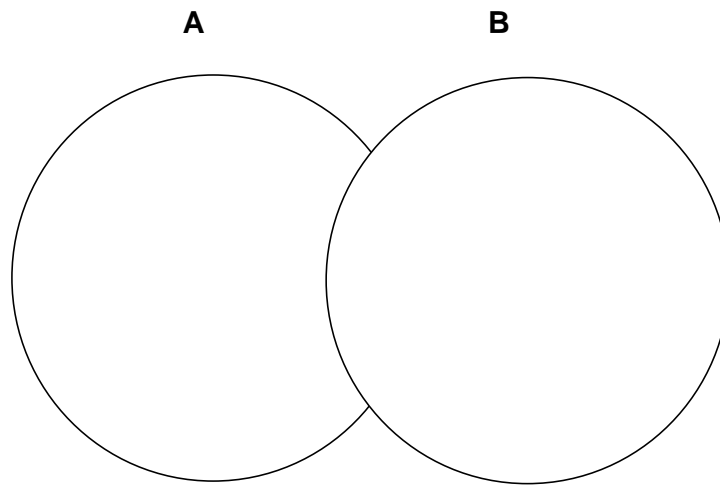
Ejemplo

```
SELECT employee_id, job_id, department_id
FROM   employees
INTERSECT
SELECT employee_id, job_id, department_id
FROM   job_history;
```

EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
176	SA_REP	80

El empleado 200 ya no forma parte de los resultados porque el valor EMPLOYEES.DEPARTMENT_ID es diferente al valor JOB_HISTORY.DEPARTMENT_ID.

El Operador MINUS



ORACLE

15-14

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

El Operador MINUS

Utilice el operador MINUS para devolver filas devueltas por la primera consulta que no estén presentes en la segunda (la primera sentencia SELECT menos (MINUS) la segunda sentencia SELECT).

Instrucciones

- El número de columnas y los tipos de datos de las columnas seleccionadas por las sentencias SELECT deben ser idénticos en todas las sentencias SELECT utilizadas en las consultas. Los nombres de las columnas no tienen que ser idénticos.
- Todas las columnas de la cláusula WHERE deben estar en la cláusula SELECT para que el operador MINUS funcione.

El Operador MINUS

Visualice los identificadores de empleado de los empleados que no hayan cambiado de cargo ni una sola vez.

```
SELECT employee_id, job_id
FROM   employees
MINUS
SELECT employee_id, job_id
FROM   job_history;
```

EMPLOYEE_ID	JOB_ID
100	AD_PRES
101	AD_VP
102	AD_VP
103	IT_PROG
...	
201	MK_MAN
202	MK_REP
205	AC_MGR
206	AC_ACCOUNT

18 rows selected.

ORACLE

El Operador MINUS (continuación)

En el ejemplo de la transparencia, los identificadores de empleado y de cargo de la tabla JOB_HISTORY se restan a los de la tabla EMPLOYEES. El juego de resultados muestra los empleados restantes tras la resta; se representan con filas que existen en la tabla EMPLOYEES pero que no existen en la tabla JOB_HISTORY. Son los registros de los empleados que no han cambiado de cargo ni una sola vez.

Instrucciones para Operadores SET

- Las expresiones de las listas **SELECT** deben coincidir en número y tipo de datos.
- Se pueden utilizar paréntesis para modificar la secuencia de ejecución.
- La cláusula **ORDER BY**:
 - Puede aparecer sólo justo al final de la sentencia.
 - Aceptará el nombre de columna, los alias de la primera sentencia **SELECT** o la notación posicional.

ORACLE

15-16

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

Instrucciones para Operadores SET

- Las expresiones de las listas **SELECT** de las consultas deben coincidir en número y tipo de datos. Las consultas que utilizan los operadores **SET UNION**, **UNION ALL**, **INTERSECT** y **MINUS** en su cláusula **WHERE** deben tener el mismo número y tipo de columnas en su lista **SELECT**. Por ejemplo:

```
SELECT employee_id, department_id
FROM   employees
WHERE  (employee_id, department_id)
       IN (SELECT employee_id, department_id
           FROM   employees
           UNION
           SELECT employee_id, department_id
           FROM   job_history);
```
- La cláusula **ORDER BY**:
 - Puede aparecer sólo justo al final de la sentencia.
 - Aceptará un nombre de columna, un alias o la notación posicional.
- Si se utiliza el nombre o alias de columna en una cláusula **ORDER BY**, debe ser el de la primera lista **SELECT**.
- Los operadores **SET** se pueden utilizar en subconsultas.

Oracle Server y los Operadores SET

- Las filas duplicadas se eliminan automáticamente excepto en UNION ALL.
- Los nombres de columna de la primera consulta aparecen en el resultado.
- El resultado se ordena en orden ascendente por defecto excepto en UNION ALL.

ORACLE

15-17

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

Oracle Server y los Operadores SET

Cuando una consulta utiliza operadores SET, Oracle Server elimina las filas duplicadas excepto en el caso del operador UNION ALL. Los nombres de columna de la salida los decide la lista de columnas de la primera sentencia SELECT. Por defecto, el resultado se ordena en orden ascendente por la primera columna de la cláusula SELECT.

Las expresiones correspondientes de las listas SELECT de las consultas componentes de una consulta compuesta deben coincidir en número y tipo de datos. Si las consultas componentes seleccionan datos de caracteres, el tipo de datos de los valores de devolución se determinan como se indica a continuación:

- Si ambas consultas seleccionan valores del tipo de datos CHAR, los valores devueltos tienen el tipo de datos CHAR.
- Si cualquiera de las consultas o ambas seleccionan valores del tipo de datos VARCHAR2, los valores devueltos tienen el tipo de datos VARCHAR2.

Coincidencia de las Sentencias SELECT

Mediante el operador UNION, visualice el identificador de departamento, la ubicación y la fecha de contratación de todos los empleados.

```
SELECT department_id, TO_NUMBER(null)
      location, hire_date
FROM   employees
UNION
SELECT department_id, location_id,  TO_DATE(null)
FROM   departments;
```

DEPARTMENT_ID	LOCATION	HIRE_DATE
10	1700	
10		17-SEP-87
20	1800	
20		17-FEB-96
...		
110	1700	
110		07-JUN-94
190	1700	
		24-MAY-99

27 rows selected.

ORACLE

Coincidencia de las Sentencias SELECT

Como las expresiones de las listas SELECT de las consultas deben coincidir en número, puede utilizar las columnas ficticias y las funciones de conversión de tipos de datos para cumplir esta regla. En la transparencia, se pone el nombre location como cabecera de columna ficticia. La función TO_NUMBER se utiliza en la primera consulta para hacer coincidir el tipo de datos NUMBER de la columna LOCATION_ID recuperada por la segunda consulta. De forma parecida, la función TO_DATE de la segunda consulta se utiliza para hacer coincidir el tipo de datos DATE de la columna HIRE_DATE recuperada por la primera consulta.

Coincidencia de las Sentencias SELECT

- Mediante el operador UNION, visualice los identificadores de empleado y de cargo, y el salario de todos los empleados.

```
SELECT employee_id, job_id,salary
FROM   employees
UNION
SELECT employee_id, job_id,0
FROM   job_history;
```

EMPLOYEE_ID	JOB_ID	SALARY
100	AD_PRES	24000
101	AC_ACCOUNT	0
101	AC_MGR	0
...		
205	AC_MGR	12000
206	AC_ACCOUNT	8300

30 rows selected.

ORACLE

Coincidencia de las Sentencias SELECT: Ejemplo

Las tablas EMPLOYEES y JOB_HISTORY tienen varias columnas en común; por ejemplo: EMPLOYEE_ID, JOB_ID y DEPARTMENT_ID. Pero, ¿qué sucede si desea que la consulta muestre EMPLOYEE_ID, JOB_ID y SALARY mediante el operador UNION, sabiendo que el salario existe sólo en la tabla EMPLOYEES?

El código de ejemplo de la transparencia coincide con las columnas EMPLOYEE_ID y JOB_ID de las tablas EMPLOYEES y JOB_HISTORY. Se agrega un valor literal de 0 a la sentencia JOB_HISTORY SELECT para hacer coincidir con la columna numérica SALARY de la sentencia EMPLOYEES SELECT.

En los resultados anteriores, cada fila de la salida que se corresponde con un registro de la tabla JOB_HISTORY contiene un 0 en la columna SALARY.

Control del Orden de las Filas

Cree una frase en inglés mediante dos operadores UNION.

```
COLUMN a_dummy NOPRINT
SELECT 'sing' AS "My dream", 3 a_dummy
FROM dual
UNION
SELECT 'I'd like to teach', 1
FROM dual
UNION
SELECT 'the world to', 2
FROM dual
ORDER BY 2;
```

My dream
I'd like to teach
the world to
sing

ORACLE

15-20

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

Control del Orden de las Filas

Por defecto, el resultado se ordena en orden ascendente por la primera columna. Se puede utilizar la cláusula `ORDER BY` para cambiarlo.

Uso de `ORDER BY` para Ordenar las Filas

Se puede utilizar la cláusula `ORDER BY` sólo una vez en una consulta compuesta. Si se utiliza, la cláusula `ORDER BY` se debe colocar al final de la consulta. La cláusula `ORDER BY` acepta el nombre de columna, un alias o la notación posicional. Sin la cláusula `ORDER BY`, el código de ejemplo de la transparencia produce la siguiente salida en orden alfabético por la primera columna:

My dream
I'd like to teach
sing
the world to

Note: Piense en una consulta compuesta en la que el operador `UNION` se utilice más de una vez. En este caso, la cláusula `ORDER BY` sólo puede utilizar posiciones en lugar de expresiones explícitas.

Resumen

En esta lección, debería haber aprendido a:

- **Utilizar UNION para devolver todas las filas distintas**
- **Utilizar UNION ALL para devolver todas las filas, incluidas las duplicadas**
- **Utilizar INTERSECT para devolver todas las filas compartidas por ambas consultas**
- **Utilizar MINUS para devolver todas las filas distintas de la primera consulta pero no de la segunda**
- **Utilizar ORDER BY sólo justo al final de la sentencia**

ORACLE

15-21

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

Resumen

- El operador UNION devuelve todas las filas seleccionadas por cualquiera de las consultas. Utilice el operador UNION para devolver todas las filas de varias tablas y eliminar las filas duplicadas.
- Utilice el operador UNION ALL para devolver todas las filas de varias consultas. A diferencia del operador UNION, no se eliminan las filas duplicadas y el resultado no se ordena por defecto.
- Utilice el operador INTERSECT para devolver todas las filas comunes de varias consultas.
- Utilice el operador MINUS para devolver las filas devueltas por la primera consulta que no estén presentes en la segunda.
- Recuerde utilizar la cláusula ORDER BY sólo justo al final de la sentencia compuesta.
- Asegúrese de que las expresiones correspondientes de las listas SELECT coinciden en número y en tipo de datos.

Visión General de la Práctica 15

Esta práctica abarca el uso de las funciones datetime de Oracle9i

ORACLE

15-22

Copyright © Oracle Corporation, 2001. Todos los Derechos Reservados.

Visión General de la Práctica 15

En esta práctica, escribirá consultas mediante los operadores SET.

Práctica 15

1. Enumere los identificadores de departamento para departamentos que no contienen el cargo ID ST_CLERK, mediante operadores SET.

DEPARTMENT_ID	
	10
	20
	60
	80
	90
	110
	190

7 rows selected.

2. Mediante operadores SET, visualice el identificador de país y el nombre de los países que no tengan departamentos ubicados en ellos.

CO	COUNTRY_NAME
DE	Germany

3. Cree una lista de puestos para los departamentos 10, 50 y 20, en ese orden. Visualice el identificador de cargo y de departamento, mediante operadores SET.

JOB_ID	DEPARTMENT_ID
AD_ASST	10
ST_CLERK	50
ST_MAN	50
MK_MAN	20
MK_REP	20

4. Enumere los identificadores de empleado y de cargo de los empleados que actualmente tengan el cargo que ocupaban antes de comenzar a trabajar con la compañía.

EMPLOYEE_ID	JOB_ID
176	SA_REP
200	AD_ASST

Práctica 15 (continuación)

5. Escriba una consulta compuesta que enumere:

- Apellidos e identificadores de departamento de todos los empleados de la tabla EMPLOYEES, independientemente de si pertenecen o no a algún departamento
- Identificador y nombre de departamento de todos los departamentos de la tabla DEPARTMENTS, independientemente de si tienen o no empleados trabajando en ellos.

18

Subconsultas Avanzadas

ORACLE®

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

Objetivos

Al finalizar esta lección, debería estar capacitado para:

- **Escribir una subconsulta de varias columnas**
- **Describir y explicar el comportamiento de las subconsultas cuando se recuperan valores nulos**
- **Escribir una subconsulta en una cláusula FROM**
- **Utilizar subconsultas escalares en SQL**
- **Describir los tipos de problemas que se pueden resolver con subconsultas correlacionadas**
- **Escribir subconsultas correlacionadas**
- **Actualizar y suprimir filas mediante subconsultas correlacionadas**
- **Utilizar los operadores EXISTS y NOT EXISTS**
- **Utilizar la cláusula WITH**

ORACLE

18-2

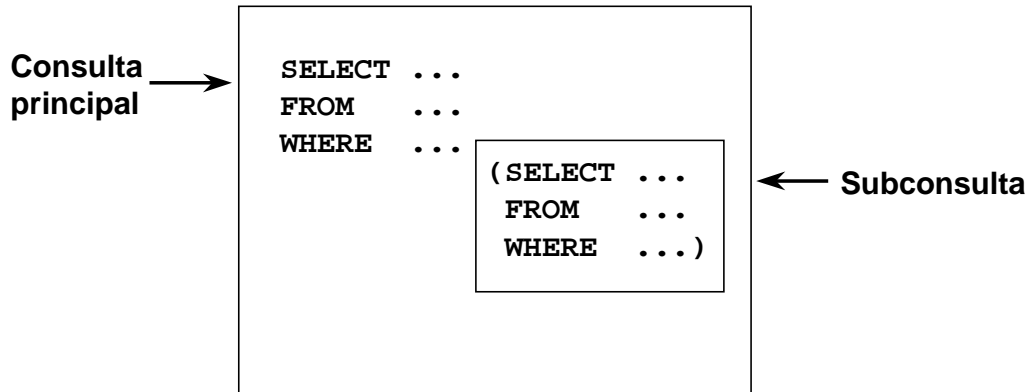
Copyright © Oracle Corporation, 2001. Todos los derechos reservados.

Objetivo de la Lección

En esta lección, aprenderá a escribir subconsultas de varias columnas y subconsultas en la cláusula FROM de una sentencia SELECT. También aprenderá a resolver problemas mediante subconsultas correlacionadas, escalares y la cláusula WITH.

¿Qué Es una Subconsulta?

Una subconsulta es una sentencia **SELECT** que está embebida en una cláusula de otra sentencia **SQL**.



ORACLE

18-3

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

¿Qué Es una Subconsulta?

Una *subconsulta* es una sentencia **SELECT** que está embebida en una cláusula de otra sentencia **SQL**, llamada sentencia principal.

La subconsulta (consulta interna) devuelve un valor que utiliza la sentencia principal. Utilizar una subconsulta anidada es equivalente a realizar dos consultas secuenciales y usar el resultado de la consulta interna como valor de búsqueda en la consulta externa (principal).

Las subconsultas se pueden utilizar para:

- Proporcionar valores para condiciones de cláusulas **WHERE**, **HAVING** y **START WITH** de sentencias **SELECT**
- Definir el juego de filas que se ha de insertar en la tabla destino de una sentencia **INSERT** o **CREATE TABLE**
- Definir el juego de filas que se ha de incluir en una vista o instantánea de una sentencia **CREATE VIEW** o **CREATE SNAPSHOT**
- Definir uno o varios valores que se han de asignar a filas existentes de una sentencia **UPDATE**
- Definir una tabla en la que va a operar una consulta que la contiene. (Esto se hace colocando la subconsulta en la cláusula **FROM**. Esto también se puede hacer en sentencias **INSERT**, **UPDATE** y **DELETE**)

Nota: Una subconsulta se evalúa una vez para toda la sentencia principal.

Subconsultas

```
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

18-4

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

Subconsultas

Puede crear sentencias potentes a partir de sentencias simples mediante subconsultas. Las subconsultas pueden resultar muy útiles si necesita seleccionar filas de una tabla con una condición que depende de los datos de la propia tabla o de otra tabla. Las subconsultas son muy útiles para escribir sentencias SQL que necesitan valores basados en uno o varios valores condicionales desconocidos.

En la sintaxis:

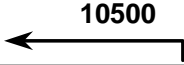
operator incluye un operador de comparación como >, = o IN

Nota: Los operadores 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. Las consultas interna y externa pueden recuperar datos de la misma tabla o de una tabla distinta.

Uso de una Subconsulta

```
SELECT last_name
FROM   employees
WHERE  salary >
      (SELECT salary
       FROM   employees
       WHERE  employee_id = 149) ;
```



LAST_NAME
King
Kochhar
De Haan
Abel
Hartstein
Higgins

6 rows selected.

ORACLE

18-5

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

Uso de una Subconsulta

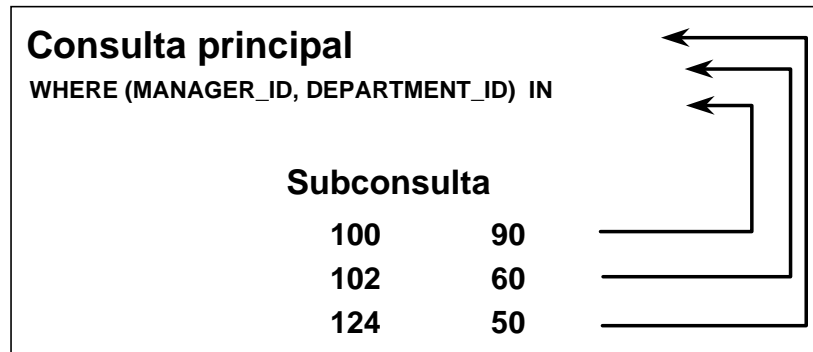
En el ejemplo de la transparencia, la consulta interna devuelve el salario del empleado número 149. La consulta externa utiliza el resultado de la interna para mostrar los nombres de todos los empleados que ganan más de esta cantidad.

Ejemplo

Muestre los nombres de todos los empleados que ganan menos del salario medio de la compañía.

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary < (SELECT AVG(salary)
                 FROM   employees) ;
```

Subconsultas de Varias Columnas



Cada fila de la consulta principal se compara con los valores de una subconsulta de varias filas y de varias columnas.

ORACLE

18-6

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

Subconsultas de Varias Columnas

Hasta ahora ha escrito subconsultas de una sola fila y subconsultas de varias filas en las que la sentencia `SELECT` interna sólo devuelve una columna, que se utiliza para evaluar la expresión de la sentencia `SELECT` principal. Si desea comparar dos o más columnas, debe escribir una cláusula `WHERE` compuesta utilizando operadores lógicos. Con subconsultas de varias columnas, puede combinar condiciones `WHERE` duplicadas en una única cláusula `WHERE`.

Sintaxis

```
SELECT  column, column, ...
FROM    table
WHERE   (column, column, ...) IN
        (SELECT column, column, ...
         FROM   table
         WHERE  condition);
```

El gráfico de la transparencia muestra que los valores de `MANAGER_ID` y `DEPARTMENT_ID` de la consulta principal se comparan con los valores de `MANAGER_ID` y `DEPARTMENT_ID` recuperados por la subconsulta. Como se compara más de una columna, el ejemplo cualifica como una subconsulta de varias columnas.

Comparaciones de Columnas

Las comparaciones de columnas en una subconsulta de varias columnas pueden ser:

- **Comparaciones entre pares**
- **Comparaciones no entre pares**

ORACLE

18-7

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

Comparaciones entre Pares frente a Comparaciones No entre Pares

Las comparaciones de columnas de una subconsulta de varias columnas pueden ser comparaciones entre pares o comparaciones no entre pares.

En el ejemplo de la siguiente transparencia, se ejecutó una comparación entre pares en la cláusula WHERE. Cada posible fila de la sentencia SELECT debe tener *tanto* la misma columna MANAGER_ID como el mismo DEPARTMENT_ID que el empleado con EMPLOYEE_ID 178 o 174.

Una subconsulta de varias columnas también puede ser una comparación no entre pares. En este tipo de comparaciones, cada una de las columnas de la cláusula WHERE de la sentencia SELECT principal se comparan individualmente con varios valores recuperados por la sentencia SELECT interna. Las columnas individuales pueden coincidir con cualquiera de los valores recuperados por la sentencia SELECT interna. Pero, colectivamente, se deben satisfacer todas las condiciones de la sentencia SELECT principal para que se muestre la fila. El ejemplo de la siguiente página muestra una comparación no entre pares.

Subconsulta de Comparación entre Pares

Visualice los detalles de los empleados dirigidos por el mismo director y que trabajen en el mismo departamento que los empleados cuyo `EMPLOYEE_ID` sea 178 o 174.

```
SELECT employee_id, manager_id, department_id
FROM   employees
WHERE  (manager_id, department_id) IN
      (SELECT manager_id, department_id
       FROM   employees
       WHERE  employee_id IN (178,174))
AND    employee_id NOT IN (178,174);
```

ORACLE®

18-8

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

Subconsulta de Comparación entre Pares

El ejemplo de la transparencia es de una subconsulta de varias columnas porque la subconsulta devuelve más de una columna. Compara los valores de la columna `MANAGER_ID` y de la columna `DEPARTMENT_ID` de cada fila de la tabla `EMPLOYEES` con los valores de la columna `MANAGER_ID` y de la columna `DEPARTMENT_ID` para los empleados con `EMPLOYEE_ID` 178 o 174.

En primer lugar se ejecuta la subconsulta que recupera los valores de `MANAGER_ID` y `DEPARTMENT_ID` para el empleado con `EMPLOYEE_ID` 178 o 174. Estos valores se comparan con la columna `MANAGER_ID` y la columna `DEPARTMENT_ID` de cada fila de la tabla `EMPLOYEES`. Si los valores coinciden, se muestra la fila. En la salida no se mostrarán los registros de los empleados con `EMPLOYEE_ID` 178 o 174. A continuación, se muestra la salida de la consulta de la transparencia.

EMPLOYEE_ID	MANAGER_ID	DEPARTMENT_ID
176	149	80

Subconsulta de Comparación No entre Pares

Visualice los detalles de los empleados dirigidos por el mismo director que los empleados con `EMPLOYEE_ID` 174 o 141 y que trabajen en el mismo departamento que los empleados con `EMPLOYEE_ID` 174 o 141.

```
SELECT  employee_id, manager_id, department_id
FROM    employees
WHERE   manager_id IN
        (SELECT  manager_id
         FROM    employees
         WHERE   employee_id IN (174,141))
AND     department_id IN
        (SELECT  department_id
         FROM    employees
         WHERE   employee_id IN (174,141))
AND     employee_id NOT IN(174,141);
```

ORACLE

18-9

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

Subconsulta de Comparación No entre Pares

El ejemplo muestra una comparación no entre pares de las columnas. Muestra `EMPLOYEE_ID`, `MANAGER_ID` y `DEPARTMENT_ID` de cualquier empleado cuyo identificador de director coincida con el identificador de director de los empleados con identificadores de empleado 174 o 141 y cuyo `DEPARTMENT_ID` coincida con cualquiera de los identificadores de departamento de los empleados con identificadores de empleado 174 o 141.

En primer lugar, se ejecuta la subconsulta que recupera los valores de `MANAGER_ID` para los empleados con `EMPLOYEE_ID` 174 o 141. Del mismo modo, se ejecuta la segunda subconsulta que recupera los valores de `DEPARTMENT_ID` para los empleados con `EMPLOYEE_ID` 174 o 141. Los valores recuperados de las columnas `MANAGER_ID` y `DEPARTMENT_ID` se comparan con las columnas `MANAGER_ID` y `DEPARTMENT_ID` para cada fila de la tabla `EMPLOYEES`. Se muestra el registro, si la columna `MANAGER_ID` de la fila de la tabla `EMPLOYEES` coincide con alguno de los valores de `MANAGER_ID` recuperados por la subconsulta interna y si la columna `DEPARTMENT_ID` de la fila de la tabla `EMPLOYEES` coincide con alguno de los valores de `DEPARTMENT_ID` recuperados por la segunda subconsulta. A continuación, se muestra la salida de la consulta de la transparencia.

EMPLOYEE_ID	MANAGER_ID	DEPARTMENT_ID
142	124	50
143	124	50
144	124	50
176	149	80

Uso de una Subconsulta en la Cláusula FROM

```
SELECT  a.last_name, a.salary,  
        a.department_id, b.salavg  
FROM    employees a, (SELECT  department_id,  
                        AVG(salary) salavg  
                        FROM    employees  
                        GROUP BY department_id) b  
WHERE   a.department_id = b.department_id  
AND     a.salary > b.salavg;
```

LAST_NAME	SALARY	DEPARTMENT_ID	SALAVG
Hartstein	13000	20	9500
Mourgos	5800	50	3500
Hunold	9000	60	6400
Zlotkey	10500	80	10033.3333
Abel	11000	80	10033.3333
King	24000	90	19333.3333
Higgins	12000	110	10150

7 rows selected.

ORACLE

18-10

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

Uso de una Subconsulta en la Cláusula FROM

Puede utilizar una subconsulta en la cláusula FROM de una sentencia SELECT, que es muy similar al modo en que se usan las vistas. Una subconsulta en la cláusula FROM de una sentencia SELECT también se denomina *vista en línea*. Una subconsulta de la cláusula FROM de una sentencia SELECT define un origen de datos para dicha sentencia SELECT en particular y sólo para dicha sentencia SELECT. El ejemplo de la transparencia muestra apellidos, salarios, números de departamento y salarios medios para todos los empleados que ganan más que el salario medio de su departamento. La subconsulta de la cláusula FROM se llama b y la consulta externa hace referencia a la columna SALAVG que utiliza este alias.

Expresiones de Subconsulta Escalar

- Una expresión de subconsulta escalar es una subconsulta que devuelve exactamente un valor de columna de una fila.
- Las subconsultas escalares se soportaban en Oracle8i sólo en un juego limitado de casos, como, por ejemplo:
 - Sentencia **SELECT** (cláusulas **FROM** y **WHERE**)
 - Lista **VALUES** de una sentencia **INSERT**
- En Oracle9i, las subconsultas escalares se pueden utilizar en:
 - Condición y expresión parte de **DECODE** y **CASE**
 - Todas las cláusulas de **SELECT** excepto **GROUP BY**

ORACLE

18-11

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

Subconsultas Escalares en SQL

Una subconsulta que devuelve exactamente un valor de columna de una fila también se denomina subconsulta escalar. Las consultas de varias columnas escritas para comparar dos o más columnas, utilizando una cláusula **WHERE** compuesta y operadores lógicos, no cualifican como subconsultas escalares.

El valor de la expresión de subconsulta escalar es el valor del elemento de lista de selección de la subconsulta. Si ésta devuelve 0 filas, el valor de la expresión de subconsulta escalar es **NULL**. Si la subconsulta devuelve más de una fila, Oracle Server devuelve un error. Oracle Server siempre ha soportado el uso de una subconsulta escalar en una sentencia **SELECT**. El uso de subconsultas escalares se ha mejorado en Oracle9i. Ahora puede utilizar subconsultas escalares en:

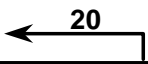
- Condición y expresión parte de **DECODE** y **CASE**
- Todas las cláusulas de **SELECT** excepto **GROUP BY**
- A la izquierda del operador en las cláusulas **SET** y **WHERE** de la sentencia **UPDATE**

Sin embargo, las subconsultas escalares no son expresiones válidas:

- Como valores por defecto para columnas y expresiones de comprobación aleatoria para agrupamientos
- En la cláusula **RETURNING** de sentencias **DML**
- Como base de un índice basado en función
- En cláusulas **GROUP BY**, restricciones **CHECK**, condiciones **WHEN**
- Cláusulas **HAVING**
- En cláusulas **START WITH** y **CONNECT BY**
- En sentencias no relacionadas con consultas, como **CREATE PROFILE**

Subconsultas Escalares: Ejemplos

Subconsultas Escalares en Expresiones

```
SELECT employee_id, last_name,  
       (CASE  
         WHEN department_id =    
           (SELECT department_id FROM departments  
            WHERE location_id = 1800)  
         THEN 'Canada' ELSE 'USA' END) location  
FROM   employees;
```

Subconsultas Escalares en la Cláusula

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

ORACLE

18-12

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

Subconsultas Escalares: Ejemplos

El primer ejemplo de la transparencia demuestra que se pueden utilizar subconsultas escalares en expresiones CASE. La consulta interna devuelve el valor 20, que es el identificador del departamento cuyo identificador de ubicación es 1800. La expresión CASE de la consulta externa utiliza el resultado de la consulta interna para mostrar el identificador de empleado, apellido y el valor Canada o USA, dependiendo de si el identificador de departamento del registro recuperado por la consulta externa es 20 o no.

A continuación se muestra el resultado del ejemplo anterior:

EMPLOYEE_ID	LAST_NAME	LOCATI
100	King	USA
101	Kochhar	USA
102	De Haan	USA
...		
201	Hartstein	Canada
202	Fay	Canada
205	Higgins	USA
206	Gietz	USA

20 rows selected.

Subconsultas Escalares: Ejemplos (continuación)

El segundo ejemplo de la transparencia demuestra que se pueden utilizar subconsultas escalares en la cláusula ORDER BY. El ejemplo ordena la salida basada en DEPARTMENT_NAME haciendo coincidir DEPARTMENT_ID de la tabla EMPLOYEES con DEPARTMENT_ID de la tabla DEPARTMENTS. Esta comparación se realiza en una subconsulta escalar en la cláusula ORDER BY. A continuación se muestra el resultado del segundo ejemplo:

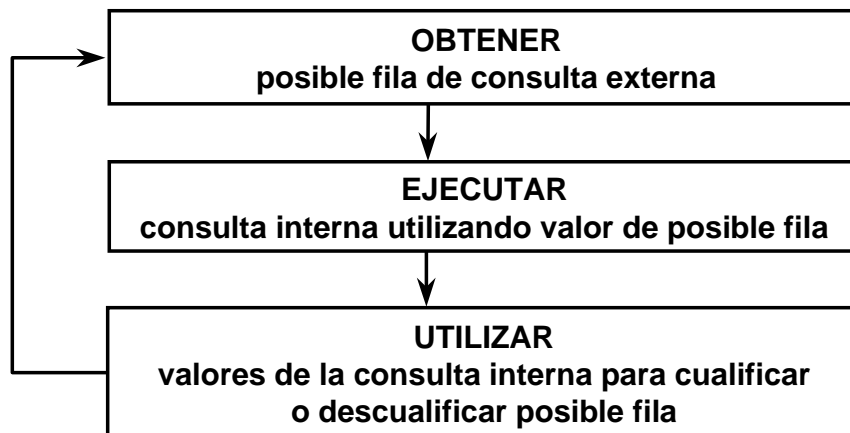
EMPLOYEE_ID	LAST_NAME
205	Higgins
206	Gietz
200	Whalen
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst
107	Lorentz
201	Hartstein
202	Fay
149	Zlotkey
176	Taylor
174	Abel
EMPLOYEE_ID	LAST_NAME
124	Mourgos
141	Rajs
142	Davies
143	Matos
144	Vargas
178	Grant

20 rows selected.

El segundo ejemplo utiliza una subconsulta correlacionada. En una subconsulta correlacionada, la subconsulta hace referencia a una columna de una tabla a la que se hace referencia en la sentencia principal. Las subconsultas correlacionadas se explican más adelante en esta lección.

Subconsultas Correlacionadas

Las subconsultas correlacionadas se utilizan para el procesamiento fila a fila. Cada subconsulta se ejecuta una vez para cada fila de la consulta externa.



ORACLE

18-14

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

Subconsultas Correlacionadas

Oracle Server realiza una subconsulta correlacionada cuando la subconsulta hace referencia a una columna de una tabla a la que se hace referencia en la sentencia principal. La subconsulta correlacionada se evalúa una vez para cada fila que procesa la sentencia principal. Ésta puede ser una sentencia `SELECT`, `UPDATE` o `DELETE`.

Subconsultas Anidadas frente a Subconsultas Correlacionadas

Con una subconsulta anidada normal, la consulta `SELECT` interna se ejecuta en primer lugar y una sola vez para devolver valores que utilizará la consulta principal. Sin embargo, una subconsulta correlacionada se ejecuta una vez para cada posible fila considerada por la consulta externa. En otras palabras, la consulta interna está controlada por la externa.

Ejecución de una Subconsulta Anidada

- La consulta interna se ejecuta primero y busca un valor.
- La consulta externa se ejecuta una vez mediante el valor de la consulta interna.

Ejecución de una Subconsulta Correlacionada

- Obtiene una posible fila (recuperada por la consulta externa).
- Ejecuta la consulta interna mediante el valor de la posible fila.
- Utiliza los valores resultantes de la consulta interna para cualificar o descualificar la posible fila.
- Repite hasta que no queda ninguna posible fila.

Subconsultas Correlacionadas

```
SELECT column1, column2, ...  
FROM   table1 outer  
WHERE  column1 operator  
        (SELECT column1, column2  
         FROM    table2  
         WHERE   expr1 =  
                outer.expr2);
```

La subconsulta hace referencia a una columna de una tabla en la consulta principal.

ORACLE

18-15

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

Subconsultas Correlacionadas (continuación)

Una subconsulta correlacionada es una forma de leer todas las filas de una tabla y comparar los valores de cada fila con los datos relacionados. Se utiliza cada vez que una subconsulta debe devolver un resultado o un juego de resultados diferentes para cada posible fila considerada por la consulta principal. En otras palabras, utilice una subconsulta correlacionada para responder a una pregunta de varias partes cuya respuesta depende del valor de cada fila procesada por la sentencia principal.


Oracle Server realiza una subconsulta correlacionada cuando la subconsulta hace referencia a una columna de una tabla en la consulta principal.

Nota: Puede utilizar los operadores ANY y ALL en una subconsulta correlacionada.

Uso de Subconsultas Correlacionadas

Busque todos los empleados que ganan más del salario medio en su departamento.

```
SELECT last_name, salary, department_id
FROM   employees outer
WHERE  salary >
      (SELECT AVG(salary)
       FROM   employees
       WHERE  department_id =
             outer.department_id) ;
```



Cada vez que se procesa una fila de la consulta externa, se evalúa la consulta interna.

ORACLE

18-16

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

Uso de Subconsultas Correlacionadas

El ejemplo de la transparencia determina qué empleados ganan más del salario medio de su departamento. En este caso, la subconsulta correlacionada calcula específicamente el salario medio para cada departamento.

Como tanto la consulta externa como la interna utilizan la tabla EMPLOYEES de la cláusula FROM, se proporciona un alias a EMPLOYEES en la sentencia SELECT externa, para mayor claridad. Este alias no solo hace que la sentencia SELECT sea más legible, sino que, sin él, la consulta no funcionaría correctamente, pues la sentencia interna no podría distinguir la columna de la tabla interna de la columna de la tabla externa.

Uso de Subconsultas Correlacionadas

Visualice detalles de los empleados que han cambiado de cargo al menos dos veces.

```
SELECT e.employee_id, last_name,e.job_id
FROM   employees e
WHERE  2 <= (SELECT COUNT(*)
              FROM   job_history
              WHERE  employee_id = e.employee_id);
```

EMPLOYEE_ID	LAST_NAME	JOB_ID
101	Kochhar	AD_VP
176	Taylor	SA_REP
200	Whalen	AD_ASST

ORACLE

18-17

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

Uso de Subconsultas Correlacionadas

El ejemplo de la transparencia muestra los detalles de los empleados que han cambiado de cargo al menos dos veces. Oracle Server evalúa una subconsulta correlacionada como se indica a continuación:

1. Selecciona una fila de la tabla especificada en la consulta externa. Será la posible fila actual.
2. Almacena el valor de la columna a la que se hace referencia en la subconsulta de esta posible fila. (En el ejemplo de la transparencia, la columna a la que se hace referencia en la subconsulta es E.EMPLOYEE_ID).
3. Realiza la subconsulta con su condición mediante la referencia al valor de la posible fila de la consulta externa. (En el ejemplo de la transparencia, la función de grupo COUNT(*) se evalúa basándose en el valor de la columna E.EMPLOYEE_ID obtenido en el paso 2).
4. Evalúa la cláusula WHERE de la consulta externa basándose en los resultados de la subconsulta realizada en el paso 3. Esto determina si la posible fila se selecciona para la salida. (En el ejemplo, el número de veces que ha cambiado de cargo un empleado, evaluado por la subconsulta, se compara con 2 en la cláusula WHERE de la consulta externa. Si se satisface la condición, se muestra el registro de este empleado).
5. Repite el procedimiento para la siguiente posible fila de la tabla y así sucesivamente, hasta que se hayan procesado todas las filas de la tabla.

La correlación se establece mediante un elemento de la consulta externa en la subconsulta. En este ejemplo, la correlación se establece con la sentencia EMPLOYEE_ID = E.EMPLOYEE_ID en la que compara EMPLOYEE_ID de la tabla de la subconsulta con EMPLOYEE_ID de la tabla de la consulta externa.

Introduction to Oracle9i: SQL 18-17

Uso del Operador EXISTS

- El operador **EXISTS** comprueba la existencia de filas en el juego de resultados de la subconsulta.
- Si se encuentra un valor de fila de la subconsulta:
 - La búsqueda no continúa en la consulta interna.
 - Se señala a la condición como **TRUE**.
- Si no se encuentra un valor de fila de la subconsulta:
 - Se señala a la condición como **FALSE**.
 - La búsqueda continúa en la consulta interna.

ORACLE

18-18

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

El Operador EXISTS

Con sentencias **SELECT** anidadas, todos los operadores lógicos son válidos. Además, puede utilizar el operador **EXISTS**. Este operador se utiliza con frecuencia con subconsultas correlacionadas para probar si existe un valor que la consulta externa haya recuperado en el juego de resultados de los valores recuperados por la consulta interna. Si la subconsulta devuelve al menos una fila, el operador devuelve **TRUE**. Si el valor no existe, devuelve **FALSE**. De la misma forma, **NOT EXISTS** prueba si un valor que la consulta externa haya recuperado no es parte del juego de resultados de los valores recuperados por la consulta interna.

Uso del Operador EXISTS

Busque los empleados que tengan al menos una persona que les informe.

```
SELECT employee_id, last_name, job_id, department_id
FROM   employees outer
WHERE  EXISTS ( SELECT 'X'
                  FROM   employees
                  WHERE  manager_id =
                        outer.employee_id);
```

EMPLOYEE_ID	LAST_NAME	JOB_ID	DEPARTMENT_ID
100	King	AD_PRES	90
101	Kochhar	AD_VP	90
102	De Haan	AD_VP	90
103	Hunold	IT_PROG	60
124	Mourgos	ST_MAN	50
149	Zlotkey	SA_MAN	80
201	Hartstein	MK_MAN	20
205	Higgins	AC_MGR	110

8 rows selected.

ORACLE

18-19

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

Uso del Operador EXISTS

El operador EXISTS asegura que la búsqueda en la consulta interna no continúa si al menos una coincidencia para el número de director y empleado se encuentra con la condición:

```
WHERE manager_id = outer.employee_id.
```

Observe que no es necesario que la consulta SELECT interna devuelva un valor específico, por lo que se puede seleccionar una constante. Desde el punto de vista del rendimiento, es más rápido seleccionar una constante que una columna.

Nota: Tener EMPLOYEE_ID en la cláusula SELECT de la consulta interna produce una exploración de la tabla para dicha columna. Sustituirla por el literal X, o cualquier constante, mejora el rendimiento. Esto es más eficaz que utilizar el operador IN.

Se puede utilizar una construcción IN como alternativa para un operador EXISTS, como se muestra en el siguiente ejemplo:

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

Uso del Operador NOT EXISTS

Busque todos los departamentos que no tengan empleados.

```
SELECT department_id, department_name
FROM departments d
WHERE NOT EXISTS (SELECT 'X'
                  FROM employees
                  WHERE department_id
                    = d.department_id);
```

DEPARTMENT_ID	DEPARTMENT_NAME
190	Contracting

ORACLE

18-20

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

Uso del Operador NOT EXISTS

Solución Alternativa

Se puede utilizar una construcción NOT IN como alternativa para un operador NOT EXISTS, como se muestra en el siguiente ejemplo:

```
SELECT department_id, department_name
FROM departments
WHERE department_id NOT IN (SELECT department_id
                           FROM employees);
```

no rows selected

Sin embargo, NOT IN se evalúa en FALSE si algún miembro del juego es un valor NULL. Por lo tanto, la consulta no devolverá ninguna fila aunque haya filas en la tabla de departamentos que satisfagan la condición WHERE.

UPDATE Correlacionado

```
UPDATE table1 alias1
SET    column = (SELECT expression
                    FROM    table2 alias2
                    WHERE   alias1.column =
                            alias2.column);
```

Utilice una subconsulta correlacionada para actualizar las filas de una tabla basadas en las filas de otra tabla.

ORACLE

18-21

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

UPDATE Correlacionado

En el caso de la sentencia UPDATE, puede utilizar una subconsulta correlacionada para actualizar las filas de una tabla basadas en las filas de otra tabla.

UPDATE Correlacionado

- Desnormalice la tabla `EMPLOYEES` agregando una columna que almacene el nombre de departamento.
- Rellene la tabla mediante una actualización correlacionada.

```
ALTER TABLE employees
ADD(department_name VARCHAR2(14));
```

```
UPDATE employees e
SET    department_name =
        (SELECT department_name
         FROM   departments d
         WHERE  e.department_id = d.department_id);
```

ORACLE

18-22

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

UPDATE Correlacionado (continuación)

Para desnormalizar la tabla `EMPLOYEES`, el ejemplo de la transparencia agrega una columna que almacena el nombre de departamento y, a continuación, rellena la tabla mediante una actualización correlacionada.

Aquí se muestra otro ejemplo de una actualización correlacionada.

Exposición del Problema

Utilice una subconsulta correlacionada para actualizar las filas de la tabla `EMPLOYEES` basadas en filas de la tabla `REWARDS`:

```
UPDATE employees
SET    salary = (SELECT employees.salary + rewards.pay_raise
                 FROM   rewards
                 WHERE  employee_id = employees.employee_id
                 AND    payraise_date =
                        (SELECT MAX(payraise_date)
                         FROM   rewards
                         WHERE  employee_id = employees.employee_id))
WHERE  employees.employee_id
IN     (SELECT employee_id
       FROM   rewards);
```

UPDATE Correlacionado (continuación)

Este ejemplo utiliza la tabla REWARDS, que contiene las columnas EMPLOYEE_ID, PAY_RAISE y PAYRAISE_DATE. Cada vez que un empleado consigue una subida salarial, se inserta en la tabla REWARDS un registro con los detalles del identificador de empleado, el importe de la subida salarial y la fecha de recepción de la subida salarial. La tabla REWARDS puede contener más de un registro para cada empleado. La columna PAYRAISE _DATE se utiliza para identificar la subida salarial más reciente que ha recibido un empleado.

En el ejemplo, la columna SALARY de la tabla EMPLOYEES se actualiza para reflejar la última subida salarial recibida por el empleado. Esto se hace sumando al salario actual del empleado la subida salarial correspondiente de la tabla REWARDS.

DELETE Correlacionado

```
DELETE FROM table1 alias1
WHERE column operator
      (SELECT expression
       FROM table2 alias2
       WHERE alias1.column = alias2.column);
```

Utilice una subconsulta correlacionada para suprimir las filas de una tabla basadas en filas de otra tabla.

ORACLE

18-24

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

DELETE Correlacionado

En el caso de una sentencia DELETE, puede utilizar una subconsulta correlacionada para suprimir solamente las filas que también existen en otra tabla. Si decide que sólo mantendrá los cuatro últimos registros de historial de cargo en la tabla JOB_HISTORY, cuando un empleado pase a un quinto cargo, suprimirá la fila de JOB_HISTORY más antigua buscando en la tabla JOB_HISTORY el valor MIN(START_DATE) del empleado. El siguiente código muestra el modo en que se puede realizar la operación anterior mediante un DELETE correlacionado:

```
DELETE FROM job_history JH
WHERE employee_id =
      (SELECT employee_id
       FROM employees E
       WHERE JH.employee_id = E.employee_id
       AND start_date =
            (SELECT MIN(start_date)
             FROM job_history JH
             WHERE JH.employee_id = E.employee_id)
       AND 5 > (SELECT COUNT(*)
                FROM job_history JH
                WHERE JH.employee_id = E.employee_id
                GROUP BY employee_id
                HAVING COUNT(*) >= 4));
```


DELETE Correlacionado

Utilice una subconsulta correlacionada para suprimir solamente las filas de la tabla **EMPLOYEES** que también existen en la tabla **EMP_HISTORY**.

```
DELETE FROM employees E
WHERE employee_id =
      (SELECT employee_id
       FROM   emp_history
       WHERE  employee_id = E.employee_id);
```

ORACLE

18-25

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

DELETE Correlacionado (continuación)

Ejemplo

En este ejemplo se utilizan dos tablas:

- La tabla **EMPLOYEES**, que proporciona detalles de todos los empleados actuales.
- La tabla **EMP_HISTORY**, que proporciona detalles de los empleados anteriores.

EMP_HISTORY contiene datos sobre empleados anteriores, por lo que sería erróneo que existiera el registro del mismo empleado tanto en la tabla **EMPLOYEES** como en la tabla **EMP_HISTORY**. Puede suprimir estos registros erróneos mediante la subconsulta correlacionada que se muestra en la transparencia.

La Cláusula WITH

- Con la cláusula WITH, puede utilizar el mismo bloque de consulta en una sentencia SELECT cuando se produce más de una vez dentro de una consulta compleja.
- La cláusula WITH recupera el resultado de un bloque de consulta y lo almacena en el tablespace temporal del usuario.
- La cláusula WITH mejora el rendimiento.

ORACLE

18-26

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

La Cláusula WITH

Con la cláusula WITH, puede definir un bloque de consulta antes de utilizarlo en una consulta. La cláusula WITH (conocida formalmente como `subquery_factoring_clause`) le permite reutilizar el mismo bloque de consulta en una sentencia SELECT cuando se produce más de una vez dentro de una consulta compleja. Esto resulta particularmente útil cuando una consulta tiene muchas referencias al mismo bloque de consulta y hay uniones y agregaciones.

Mediante la cláusula WITH, puede reutilizar la misma consulta si es costoso evaluar el bloque de consulta y se produce más de una vez dentro de una consulta compleja. Con la cláusula WITH, Oracle Server recupera el resultado de un bloque de consulta y lo almacena en el tablespace temporal del usuario. Esto puede mejorar el rendimiento.

Ventajas de la Cláusula WITH

- Facilita la lectura de la consulta.
- Evalúa una cláusula una sola vez, aunque aparezca varias veces en la consulta, y por lo tanto mejora el rendimiento.

Cláusula WITH: Ejemplo

Mientras utiliza la cláusula WITH, escriba una consulta que muestre el nombre de departamento y los salarios totales de los departamentos cuyos salarios totales sean mayores que el salario medio de los departamentos.

ORACLE

18-27

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

Cláusula WITH: Ejemplo

El problema de la transparencia requeriría los siguientes cálculos intermedios:

1. Calcule el salario total de cada departamento y almacene el resultado mediante una cláusula WITH.
2. Calcule el salario medio de los departamentos y almacene el resultado mediante una cláusula WITH.
3. Compare el salario total calculado en el primer paso con el salario medio calculado en el segundo paso. Si el salario total de un departamento determinado es mayor que el salario medio de los departamentos, muestre el nombre del departamento y el salario total del mismo.

La solución del problema anterior se proporciona en la siguiente página.

Cláusula WITH: Ejemplo

WITH

```
dept_costs AS (  
  SELECT d.department_name, SUM(e.salary) AS dept_total  
  FROM   employees e, departments d  
  WHERE  e.department_id = d.department_id  
  GROUP BY d.department_name),  
avg_cost AS (  
  SELECT SUM(dept_total)/COUNT(*) AS dept_avg  
  FROM   dept_costs)  
SELECT *  
FROM   dept_costs  
WHERE  dept_total >  
       (SELECT dept_avg  
        FROM avg_cost)  
ORDER BY department_name;
```

ORACLE

18-28

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

Cláusula WITH: Ejemplo (continuación)

El código SQL de la transparencia es un ejemplo de una situación en la que puede mejorar el rendimiento y escribir SQL de forma más simple mediante la cláusula WITH. La consulta crea los nombres de consulta DEPT_COSTS y AVG_COST y luego los utiliza en el cuerpo de la consulta principal. Internamente, la cláusula WITH se resuelve como una vista en línea o una tabla temporal. El optimizador selecciona la resolución adecuada dependiendo del costo o del beneficio de almacenar temporalmente los resultados de la cláusula WITH.

Nota: Una subconsulta de la cláusula FROM de una sentencia SELECT también se llama vista en línea.

La salida generada por el código SQL de la transparencia será:

DEPARTMENT_NAME	DEPT_TOTAL
Executive	58000
Sales	30100

Notas sobre el Uso de la Cláusula WITH

- Sólo se utiliza con sentencias SELECT.
- Un nombre de consulta está visible para todos los bloques de consulta del elemento WITH (incluidos sus bloques de subconsulta) definidos a partir de él y para el propio bloque de la consulta principal (incluidos sus bloques de subconsulta).
- Si el nombre de la consulta es el mismo que el de una tabla existente, el analizador busca al revés; el nombre del bloque de consulta tiene prioridad sobre el nombre de la tabla.
- La cláusula WITH puede contener más de una consulta. En este caso, cada consulta se separa con una coma.

Resumen

En esta lección, debería haber aprendido lo siguiente:

- **Una subconsulta de varias columnas devuelve más de una columna.**
- **Las comparaciones de varias columnas pueden ser entre pares o no entre pares.**
- **Una subconsulta de varias columnas también se puede utilizar en la cláusula `FROM` de una sentencia `SELECT`.**
- **Se han mejorado las subconsultas escalares en Oracle9i.**

ORACLE

18-29

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

Resumen

Puede utilizar subconsultas de varias columnas para combinar varias condiciones `WHERE` en una única cláusula `WHERE`. Las comparaciones de columnas en una subconsulta de varias columnas pueden ser entre pares o no entre pares.

Puede utilizar una subconsulta para definir una tabla en la que va a operar una consulta contenida.

Oracle 9i mejora los usos de subconsultas escalares, que se pueden utilizar ahora en:

- Condición y expresión parte de `DECODE` y `CASE`
- Todas las cláusulas de `SELECT` excepto `GROUP BY`
- Cláusulas `SET` y `WHERE` de la sentencia `UPDATE`

Resumen

- Las subconsultas correlacionadas son útiles cada vez que una subconsulta deba devolver un resultado diferente para cada posible fila.
- El operador `EXISTS` es un operador booleano que prueba la presencia de un valor.
- Se pueden utilizar subconsultas correlacionadas con sentencias `SELECT`, `UPDATE` y `DELETE`.
- Puede utilizar la cláusula `WITH` para usar el mismo bloque de consulta en una sentencia `SELECT` si se produce más de una vez.

ORACLE

18-30

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

Resumen (continuación)

Oracle Server realiza una subconsulta correlacionada si la subconsulta hace referencia a una columna de una tabla a la que se hace referencia en la sentencia principal. La subconsulta correlacionada se evalúa una vez por cada fila que procesa la sentencia principal. Ésta puede ser una sentencia `SELECT`, `UPDATE` o `DELETE`. Con la cláusula `WITH`, puede reutilizar la misma consulta si es costoso volver a evaluar el bloque de consulta y si esto se produce más de una vez dentro de una consulta compleja.

Visión General de la Práctica 18

Esta práctica cubre los siguientes temas:

- Creación de subconsultas de varias columnas
- Escritura de subconsultas correlacionadas
- Uso del operador `EXISTS`
- Uso de subconsultas escalares
- Uso de la cláusula `WITH`

ORACLE

18-31

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

Visión General de la Práctica 18

En esta práctica, escribirá subconsultas de varias columnas, correlacionadas y escalares. También resolverá problemas mediante la escritura de la cláusula `WITH`.

Práctica 18

1. Escriba una consulta para mostrar el apellido, el número de departamento y el salario de cualquier empleado cuyo número de departamento y salario coincidan con el número de departamento y salario de cualquier empleado que perciba una comisión.

LAST_NAME	DEPARTMENT_ID	SALARY
Taylor	80	8600
Zlotkey	80	10500
Abel	80	11000

2. Muestre el apellido, el nombre de departamento y el salario de cualquier empleado cuyo salario y comisión coincidan con los de cualquier empleado ubicado en el identificador de ubicación 1700.

LAST_NAME	DEPARTMENT_NAME	SALARY
Whalen	Administration	4400
Gietz	Accounting	8300
Higgins	Accounting	12000
Kochhar	Executive	17000
De Haan	Executive	17000
King	Executive	24000

6 rows selected.

3. Cree una consulta para mostrar el apellido, la fecha de contratación y el salario de todos los empleados que tengan el mismo salario y la misma comisión que Kochhar.

Nota: No muestre a Kochhar en el juego de resultados.

LAST_NAME	HIRE_DATE	SALARY
De Haan	13-JAN-93	17000

4. Cree una consulta para mostrar los empleados que ganen un salario más alto que el de todos los directores de ventas (JOB_ID = 'SA_MAN'). Ordene los resultados por salario de más alto a más bajo.

LAST_NAME	JOB_ID	SALARY
King	AD_PRES	24000
Kochhar	AD_VP	17000
De Haan	AD_VP	17000
Hartstein	MK_MAN	13000
Higgins	AC_MGR	12000
Abel	SA_REP	11000

6 rows selected.

Práctica 18 (continuación)

5. Muestre los detalles del identificador de empleado, el apellido y el identificador de departamento de los empleados que vivan en ciudades cuyos nombres comiencen por *T*.

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
201	Hartstein	20
202	Fay	20

6. Escriba una consulta para buscar todos los empleados que ganen más del salario medio en sus departamentos. Muestre el apellido, el salario, el identificador de departamento y el salario medio del departamento. Ordene por salario medio. Utilice alias para las columnas recuperadas por la consulta, como se muestra en la salida de ejemplo.

ENAME	SALARY	DEPTNO	DEPT_AVG
Mourgos	5800	50	3500
Hunold	9000	60	6400
Hartstein	13000	20	9500
Abel	11000	80	10033.3333
Zlotkey	10500	80	10033.3333
Higgins	12000	110	10150
King	24000	90	19333.3333

7 rows selected.

7. Busque a todos los empleados que no sean supervisores.
a. En primer lugar, hágalo utilizando el operador `NOT EXISTS`.

LAST_NAME
Ernst
Lorentz
Rajs
Davies
Matos
Vargas
Abel
Taylor
Grant
Whalen
Fay
Gietz

12 rows selected.

- b. ¿Se puede hacer esto mediante el operador `NOT IN`? ¿Cómo?, o ¿por qué no?

Práctica 18 (continuación)

8. Escriba una consulta para mostrar los apellidos de los empleados que ganen menos que el salario medio en sus departamentos.

LAST_NAME
Kochhar
De Haan
Ernst
Lorentz
Davies
Matos
Vargas
Taylor
Fay
Gietz

10 rows selected.

9. Escriba una consulta para mostrar los apellidos de empleados que tengan uno o más compañeros de trabajo en sus departamentos con fechas de contratación posteriores pero salarios más altos.

LAST_NAME
Rajs
Davies
Matos
Vargas
Taylor

Práctica 18 (continuación)

10. Escriba una consulta para mostrar el identificador de empleado, los apellidos y los nombres de departamento de todos los empleados.

Nota: Utilice una subconsulta escalar para recuperar el nombre de departamento en la sentencia SELECT.

EMPLOYEE_ID	LAST_NAME	DEPARTMENT
205	Higgins	Accounting
206	Gietz	Accounting
200	Whalen	Administration
100	King	Executive
101	Kochhar	Executive
102	De Haan	Executive
103	Hunold	IT
104	Ernst	IT
107	Lorentz	IT
201	Hartstein	Marketing
202	Fay	Marketing
149	Zlotkey	Sales
176	Taylor	Sales
174	Abel	Sales
EMPLOYEE_ID	LAST_NAME	DEPARTMENT
124	Mourgos	Shipping
141	Rajs	Shipping
142	Davies	Shipping
143	Matos	Shipping
144	Vargas	Shipping
178	Grant	

20 rows selected.

11. Escriba una consulta para mostrar los nombres de departamento de los departamentos cuyos costos de salario total estén por encima de un octavo ($1/8$) del costo de salario total de toda la compañía. Utilice la cláusula WITH para escribir esta consulta. Denomine SUMMARY a la consulta.

DEPARTMENT_NAME	DEPT_TOTAL
Executive	58000
Sales	30100

