CAPÍTULO 4

Filtración

A veces querrá trabajar con todas las filas de una tabla, como:

- Purgar todos los datos de una tabla que se utiliza para preparar nuevos feeds de almacenamiento de datos
- Modificar todas las filas de una tabla después de agregar una nueva columna
- Recuperar todas las filas de una tabla de cola de mensajes

En casos como estos, sus declaraciones SQL no necesitarán tener una cláusula where , ya que no es necesario excluir ninguna fila de la consideración. La mayor parte del tiempo, sin embargo, querrá limitar su enfoque a un subconjunto de las filas de una tabla. Por tanto, todos los datos SQL Las declaraciones (excepto la declaración de inserción) incluyen una cláusula where opcional para alojar condiciones de filtro utilizadas para restringir el número de filas sobre las que actúa la instrucción SQL. Además, la declaración de selección incluye una cláusula de tener en la que las condiciones de filtrado pueden incluirse los correspondientes a datos agrupados. Este capítulo explora los diversos tipos de condiciones de filtro que puede emplear en las cláusulas where de seleccionar , actualizar y eliminar declaraciones; exploramos el uso de condiciones de filtro en el que tiene cláusula de una seleccionar sentencia en el Capítulo 8.

Evaluación de condición

A donde cláusula puede contener una o más condiciones, se separó por los operadores y y o . Si varias condiciones están separadas solo por el operador y , entonces todas las condiciones debe evaluarse como verdadero para que la fila se incluya en el conjunto de resultados. Considera lo siguiente

donde cláusula:

```
DONDE título = 'Cajero' Y fecha_inicio <'2007-01-01'
```

Dadas estas dos condiciones, solo los cajeros que comenzaron a trabajar para el banco antes de 2007 se incluirá (o, para verlo de otra manera, cualquier empleado que no sea cajero o comenzó a trabajar para el banco en 2007 o más tarde será eliminado de la consideración). Aunque este ejemplo usa solo dos condiciones, no importa cuántas condiciones sean en su cláusula `where` , si están separados por el operador `y` , todos deben evaluar a `Es verdadero` para que la fila se incluya en el conjunto de resultados.

re ldt W B kC

Página 2

Sin embargo, si todas las condiciones de la cláusula `where` están separadas por el operador `or` , solo una de las condiciones debe evaluarse como `verdadera` para que la fila se incluya en el conjunto de resultados. Considere las siguientes dos condiciones:

```
DONDE título = 'Cajero' O fecha_inicio <'2007-01-01'
```

Ahora hay varias formas de incluir una fila de `empleado` determinada en el conjunto de resultados:

- El empleado es un cajero y estuvo empleado antes de 2007.
- El empleado es cajero y fue contratado después del 1 de enero de 2007.
- El empleado no es un cajero pero estuvo empleado antes de 2007.

La tabla 4-1 muestra los posibles resultados para una cláusula `where` que contiene dos condiciones separados por el operador `o` .

Tabla 4-1. Evaluación de dos condiciones usando o

Resultado intermedio	Resultado final
DONDE verdadero o verdadero	Cierto
DONDE verdadero o falso	Cierto
DONDE falso o verdadero	Cierto
DONDE falso O falso	Falso

En el caso del ejemplo anterior, la única forma de excluir una fila del El conjunto de resultados es si el empleado no es un cajero y estuvo empleado el 1 de enero de 2007 o después.

Usando paréntesis

Si su donde cláusula incluye tres o más condiciones que utilizan tanto la e y o oper-
a los usuarios, debe usar paréntesis para dejar clara su intención, tanto para el servidor de la base de datos y a cualquier otra persona que lea su código. Aquí hay una cláusula where que extiende la anterior ejemplo, comprobando para asegurarse de que el empleado todavía esté empleado por el banco:

```
DONDE end_date ES NULO
  Y (título = 'Cajero' O fecha_inicio <'2007-01-01')
```

Ahora hay tres condiciones; una fila para llegar al conjunto de resultados final, la primera la condición debe evaluarse como verdadera , y la segunda o la tercera condición (o ambas) debe evaluarse como verdadero . La Tabla 4-2 muestra los posibles resultados para esta cláusula where .

Tabla 4-2. Evaluación de tres condiciones usando y, o

Resultado intermedio	Resultado final
DONDE verdadero Y (verdadero O verdadero)	Cierto
DONDE verdadero Y (verdadero O falso)	Cierto
DONDE verdadero Y (falso O verdadero)	Cierto
DONDE verdadero Y (falso O falso)	Falso

re ldt W B kC

Página 3

Resultado intermedio	Resultado final
DONDE falso Y (verdadero O verdadero)	Falso
DONDE falso Y (verdadero O falso)	Falso
DONDE falso Y (falso O verdadero)	Falso
DONDE falso Y (falso O falso)	Falso

Como puede ver, cuantas más condiciones tenga en su cláusula where , más combinación habrá naciones hay para que el servidor las evalúe. En este caso, solo tres de las ocho combinaciones las naciones dan un resultado final de verdad .

Usando el operador not

Con suerte, el ejemplo anterior de tres condiciones es bastante fácil de entender. Considerar

sin embargo, la siguiente modificación:

```
DONDE end_date ES NULO
  Y NO (título = 'Cajero' O fecha_inicio <'2007-01-01')
```

¿Viste el cambio con respecto al ejemplo anterior? Agregué el operador `not` después del `y` operador en la segunda línea. Ahora, en lugar de buscar empleados no despedidos que son cajeros o comenzaron a trabajar para el banco antes de 2007, estoy buscando empleados no despedidos que no son contadores y comenzaron a trabajar para el banco en 2007 o posterior. La Tabla 4-3 muestra los posibles resultados de este ejemplo.

Tabla 4-3. Evaluación de tres condiciones usando y, o y no

Resultado intermedio	Resultado final
DONDE verdadero Y NO (verdadero O verdadero)	Falso
DONDE verdadero Y NO (verdadero O falso)	Falso
DONDE verdadero Y NO (falso O verdadero)	Falso
DONDE verdadero Y NO (falso O falso)	Cierto
DONDE falso Y NO (verdadero O verdadero)	Falso
DONDE falso Y NO (verdadero O falso)	Falso
DONDE falso Y NO (falso O verdadero)	Falso
DONDE falso Y NO (falso O falso)	Falso

Si bien es fácil de manejar para el servidor de base de datos, generalmente es difícil para una persona evaluar una cláusula `where` que incluya el operador `not` , razón por la cual no contrarrestarlo muy a menudo. En este caso, puede reescribir la cláusula `where` para evitar usar la `no` operador:

```
DONDE end_date ES NULO
  AND title! = 'Teller' AND start_date> = '2007-01-01'
```


re ldt W B kC

Página 4

Aunque estoy seguro de que el servidor no tiene ninguna preferencia, probablemente tenga una tiempo entendiendo esta versión de la cláusula `where` .

Construyendo una condición

Ahora que ha visto cómo el servidor evalúa múltiples condiciones, demos un paso retroceda y observe lo que comprende una sola condición. Una condición se compone de uno o más expresiones junto con uno o más operadores. Una expresión puede ser cualquiera de los siguiendo:

- Un número
- Una columna en una tabla o vista
- Una cadena literal, como 'Teller'
- Una función incorporada, como concat ('Aprendizaje', '', 'SQL')
- Una subconsulta
- Una lista de expresiones, como ('Teller', 'Head Teller', 'Operations Manager')

Los operadores utilizados dentro de las condiciones incluyen:

- los operadores de comparación, como = , ! = , < , > , <> , COMO , EN , y ENTRE
- Operadores aritméticos, como + , - , * y /

La siguiente sección demuestra cómo se pueden combinar estas expresiones y eradores para fabricar los diversos tipos de condiciones.

Tipos de condición

Hay muchas formas diferentes de filtrar los datos no deseados. Puedes buscar específicos valores, conjuntos de valores o rangos de valores para incluir o excluir, o puede utilizar varios técnicas de búsqueda de patrones para buscar coincidencias parciales cuando se trata de datos de cadena. Las siguientes cuatro subsecciones exploran cada uno de estos tipos de condiciones en detalle.

Condiciones de igualdad

Un gran porcentaje de las condiciones de filtro que escriba o con las que se encuentre será del formulario ' *columna = expresión* ' como en:

```
title = 'Cajero'
fed_id = '111-11-1111'
cantidad = 375.25
dept_id = (SELECCIONE dept_id DEL departamento DONDE nombre = 'Préstamos')
```

Condiciones como estas se denominan condiciones de igualdad porque equivalen a una presión a otro. Los primeros tres ejemplos equiparan una columna a un literal (dos cadenas y un número), y el cuarto ejemplo equipara una columna al valor devuelto por

re ldt W B kC

una subconsulta. La siguiente consulta utiliza dos condiciones de igualdad; uno en la cláusula on (un condición de unión), y el otro en la cláusula where (una condición de filtro):

```
mysql> SELECT pt.name product_type, p.name product
    -> FROM producto p INNER JOIN product_type pt
    -> EN p.product_type_cd = pt.product_type_cd
    -> DONDE pt.name = 'Cuentas de cliente';
+ ------------------ + ------------------------ +
| product_type | producto |
+ ------------------ + ------------------------ +
| Cuentas de clientes | certificado de depósito |
| Cuentas de clientes | cuenta corriente |
| Cuentas de clientes | cuenta del mercado monetario |
| Cuentas de clientes | cuenta de ahorros |
+ ------------------ + ------------------------ +
4 filas en conjunto (0.08 seg)
```

Esta consulta muestra todos los productos que son tipos de cuentas de clientes.

Condiciones de desigualdad

Otro tipo de condición bastante común es la condición de desigualdad, que afirma que dos expresiones no son iguales. Aquí está la consulta anterior con la condición de filtro en el donde la cláusula cambió a una condición de desigualdad:

```
mysql> SELECT pt.name product_type, p.name product
    -> FROM producto p INNER JOIN product_type pt
    -> EN p.product_type_cd = pt.product_type_cd
    -> DONDE pt.name <> 'Cuentas de cliente';
+ ------------------------------ + ----------------- -------- +
| tipo de producto | producto |
+ ------------------------------ + ----------------- -------- +
| Préstamos para personas y empresas | auto préstamo |
| Préstamos para personas y empresas | línea de crédito comercial |
| Préstamos para personas y empresas | hipoteca de la casa |
| Préstamos para personas y empresas | préstamos para pequeñas empresas |
+ ------------------------------ + ----------------- -------- +
4 filas en conjunto (0,00 seg)
```

Esta consulta muestra todos los productos que no son tipos de cuentas de clientes. Al construir condiciones de desigualdad, puede optar por utilizar el operador ! = o <> .

Modificación de datos mediante condiciones de igualdad

Las condiciones de igualdad / desigualdad se utilizan comúnmente al modificar datos. Por ejemplo, digamos que el banco tiene la política de eliminar filas de cuentas antiguas una vez al año. Tu La tarea es eliminar filas de la tabla de cuentas que se cerraron en 2002. Aquí hay una forma para abordarlo:

```
BORRAR DE la cuenta
DONDE estado = 'CERRADO' Y AÑO (fecha_cerrada) = 2002;
```

Esta declaración incluye dos condiciones de igualdad: una para encontrar solo cuentas cerradas y otro para verificar las cuentas cerradas en 2002.

re ldt W B kC

Página 6

Al crear ejemplos de declaraciones de eliminación y actualización , trato de escribir cada declaración de manera que no se modifiquen filas. De esa manera, cuando tu ejecutar las declaraciones, sus datos permanecerán sin cambios, y su La salida de las sentencias de selección siempre coincidirá con la que se muestra en este libro.

Dado que las sesiones de MySQL están en modo de confirmación automática de forma predeterminada (consulte el capítulo ter 12), no podrá revertir (deshacer) los cambios realizados en los datos de ejemplo si una de mis declaraciones modificó los datos. Puedes, por supuesto, haga lo que quiera con los datos de ejemplo, incluyendo wip- limpiarlo y volver a ejecutar los scripts que he proporcionado, pero trato de dejar está intacto.

Condiciones de rango

Además de comprobar que una expresión es igual (o no igual a) otra expresión, puede crear condiciones que comprueben si una expresión se encuentra dentro de un rango determinado. Este tipo de condición es común cuando se trabaja con datos numéricos o temporales. Estafa- Considere la siguiente consulta:

```
mysql> SELECT emp_id, fname, lname, start_date
    -> DESDE empleado
    -> DONDE fecha_inicio <'2007-01-01';
+ -------- + --------- + ----------- + ------------ +
| emp_id | fname | lname | fecha_inicio |
+ -------- + --------- + ----------- + ------------ +
| 1 | Michael | Smith | 2005-06-22 |
| 2 | Susan | Barker | 2006-09-12 |
| 3 | Robert | Tyler | 2005-02-09 |
| 4 | Susan | Hawthorne | 2006-04-24 |
| 8 | Sarah | Parker | 2006-12-02 |
| 9 | Jane | Grossman | 2006-05-03 |
| 10 | Paula | Roberts | 2006-07-27 |
| 11 | Thomas | Ziegler | 2004-10-23 |
| 13 | John | Blake | 2004-05-11 |
| 14 | Cindy | Mason | 2006-08-09 |
| 16 | Theresa | Markham | 2005-03-15 |
| 17 | Beth | Fowler | 2006-06-29 |
| 18 | Rick | Tulman | 2006-12-12 |
+ -------- + --------- + ----------- + ------------ +
13 filas en conjunto (0,15 seg)
```

Esta consulta busca todos los empleados contratados antes de 2007. Además de especificar un límite superior para la fecha de inicio, es posible que también desee especificar un rango inferior para la fecha de inicio:

```
mysql> SELECT emp_id, fname, lname, start_date
    -> DESDE empleado
    -> DONDE fecha_inicio <'2007-01-01'
    -> Y fecha_inicio> = '2005-01-01';
```

```
+ -------- + --------- + ----------- + ------------ +
| emp_id | fname | lname | fecha_inicio |
+ -------- + --------- + ----------- + ------------ +
```


re ldt W B kC

Página 7

```
| 1 | Michael | Smith | 2005-06-22 |
| 2 | Susan | Barker | 2006-09-12 |
| 3 | Robert | Tyler | 2005-02-09 |
| 4 | Susan | Hawthorne | 2006-04-24 |
| 8 | Sarah | Parker | 2006-12-02 |
| 9 | Jane | Grossman | 2006-05-03 |
| 10 | Paula | Roberts | 2006-07-27 |
| 14 | Cindy | Mason | 2006-08-09 |
| 16 | Theresa | Markham | 2005-03-15 |
| 17 | Beth | Fowler | 2006-06-29 |
| 18 | Rick | Tulman | 2006-12-12 |
+ -------- + --------- + ----------- + ------------ +
11 filas en conjunto (0,00 seg)
```

Esta versión de la consulta recupera todos los empleados contratados en 2005 o 2006.

El operador entre

Cuando tenga un límite superior e inferior para su rango, puede optar por utilizar una condición única que utiliza el operador intermedio en lugar de utilizar dos condiciones, como en:

```
mysql> SELECT emp_id, fname, lname, start_date
    -> DESDE empleado
    -> DONDE fecha_inicio ENTRE '2005-01-01' Y '2007-01-01';
+ -------- + --------- + ----------- + ------------ +
| emp_id | fname | lname | fecha_inicio |
+ -------- + --------- + ----------- + ------------ +
| 1 | Michael | Smith | 2005-06-22 |
| 2 | Susan | Barker | 2006-09-12 |
| 3 | Robert | Tyler | 2005-02-09 |
| 4 | Susan | Hawthorne | 2006-04-24 |
| 8 | Sarah | Parker | 2006-12-02 |
| 9 | Jane | Grossman | 2006-05-03 |
| 10 | Paula | Roberts | 2006-07-27 |
| 14 | Cindy | Mason | 2006-08-09 |
| 16 | Theresa | Markham | 2005-03-15 |
| 17 | Beth | Fowler | 2006-06-29 |
```

```
| 18 | Rick | Tulman | 2006-12-12 |
+ -------- + --------- + ----------- + ------------ +
11 filas en conjunto (0.03 seg)
```

Al utilizar el operador between , hay un par de cosas a tener en cuenta. Tú siempre debe especificar el límite inferior del rango primero (después entre) y el superior límite del rango segundo (después y). Esto es lo que sucede si especifica por error el límite superior primero:

```
mysql> SELECT emp_id, fname, lname, start_date
    -> DESDE empleado
    -> DONDE fecha_inicio ENTRE '2007-01-01' Y '2005-01-01';
Conjunto vacío (0,00 seg)
```

Como puede ver, no se devuelven datos. Esto se debe a que el servidor está generando dos condiciones de su condición única usando los operadores <= y > = , como en:

re ldt W B kC

Página 8

```
mysql> SELECT emp_id, fname, lname, start_date
    -> DESDE empleado
    -> DONDE fecha_inicio> = '2007-01-01'
    -> Y fecha_inicio <= '2005-01-01';
Conjunto vacío (0,00 seg)
```

Dado que es imposible tener una fecha mayor que el 1 de enero de 2007 y menor que el 1 de enero de 2005, la consulta devuelve un conjunto vacío. Esto me lleva al segundo error al usar between , que es recordar que sus límites superior e inferior son inclusive, lo que significa que los valores que proporcione se incluyen en los límites del rango. En esto caso, quiero especificar 2005-01-01 como el extremo inferior del rango y 2006-12-31 como el extremo superior, en lugar de 2007-01-01 . Aunque probablemente no hubiera empleados que comenzó a trabajar para el banco el día de Año Nuevo de 2007, es mejor especificar exactamente Lo que quieras.

Junto con las fechas, también puede crear condiciones para especificar rangos de números. Numérico los rangos son bastante fáciles de comprender, como lo demuestra lo siguiente:

```
mysql> SELECT account_id, product_cd, cust_id, avail_balance
    -> DESDE cuenta
    -> DONDE avail_balance ENTRE 3000 Y 5000;
+ ------------ + ------------ + --------- + ------------- - +
| account_id | product_cd | cust_id | avail_balance |
+ ------------ + ------------ + --------- + ------------- - +
```

```
|                  3 | discos compactos      | 1 | 3000,00 |
|                 17 | discos compactos      | 7 | 5000,00 |
|                 18 | CHK                   | 8 | 3487.19 |
+ ------------ + ------------ + --------- + ------------- - +
3 filas en conjunto (0.10 seg)
```

Se devuelven todas las cuentas con un saldo disponible de entre $ 3,000 y $ 5,000. Nuevamente, asegúrese de especificar primero la cantidad más baja.

Rangos de cadenas

Si bien los rangos de fechas y números son fáciles de entender, también puede crear condiciones que buscan rangos de cadenas, que son un poco más difíciles de visualizar. Diga, por ejemplo En general, está buscando clientes que tengan un número de seguro social que se encuentre dentro de un cierto rango. El formato de un número de seguro social es "XXX-XX-XXXX", donde X es un número del 0 al 9 y desea encontrar todos los clientes cuyo Seguro Social El número se encuentra entre "500-00-0000" y "999-99-9999". Esto es lo que dice la declaración se vería así:

```
mysql> SELECT cust_id, fed_id
    -> DEL cliente
    -> DONDE cust_type_cd = 'I'
    -> Y fed_id ENTRE '500-00-0000' Y '999-99-9999';
+ --------- + ------------- +
| cust_id | fed_id |
+ --------- + ------------- +
| 5 | 555-55-5555 |
| 6 | 666-66-6666 |
```


re ldt W B kC

Página 9

```
| 7 | 777-77-7777 |
| 8 | 888-88-8888 |
| 9 | 999-99-9999 |
+ --------- + ------------- +
5 filas en conjunto (0.01 seg)
```

Para trabajar con rangos de cadenas, necesita conocer el orden de los caracteres dentro de su juego de caracteres (el orden en el que se ordenan los caracteres dentro de un juego de caracteres se llama una colación).

Condiciones de afiliación

En algunos casos, no restringirá una expresión a un solo valor o rango de valores, sino más bien a un conjunto finito de valores. Por ejemplo, es posible que desee ubicar todos cuentas cuyo código de producto es 'CHK' , 'SAV' , 'CD' o 'MM' :

```
mysql> SELECT account_id, product_cd, cust_id, avail_balance
    -> DESDE cuenta
    -> DONDE product_cd = 'CHK' O product_cd = 'SAV'
    -> O product_cd = 'CD' O product_cd = 'MM';
+ ------------ + ------------ + --------- + ------------- - +
| account_id | product_cd | cust_id | avail_balance |
+ ------------ + ------------ + --------- + ------------- - +
| 1 | CHK | 1 | 1057.75 |
| 2 | SAV | 1 | 500,00 |
| 3 | discos compactos | 1 | 3000,00 |
| 4 | CHK | 2 | 2258.02 |
| 5 | SAV | 2 | 200,00 |
| 7 | CHK | 3 | 1057.75 |
| 8 | MM | 3 | 2212.50 |
| 10 | CHK | 4 | 534.12 |
| 11 | SAV | 4 | 767,77 |
| 12 | MM | 4 | 5487.09 |
| 13 | CHK | 5 | 2237,97 |
| 14 | CHK | 6 | 122,37 |
| 15 | discos compactos | 6 | 10000,00 |
| 17 | discos compactos | 7 | 5000,00 |
| 18 | CHK | 8 | 3487.19 |
| 19 | SAV | 8 | 387,99 |
| 21 | CHK | 9 | 125,67 |
| 22 | MM | 9 | 9345.55 |
| 23 | discos compactos | 9 | 1500,00 |
| 24 | CHK | 10 | 23575.12 |
| 28 | CHK | 12 | 38552.05 |
+ ------------ + ------------ + --------- + ------------- - +
21 filas en conjunto (0,28 seg)
```

Si bien esta cláusula where (cuatro condiciones o 'd juntas) no fue demasiado tediosa de generar, imagina si el conjunto de expresiones contuviera 10 o 20 miembros. Para estas situaciones, usted puede usar el operador in en su lugar:

```
SELECT account_id, product_cd, cust_id, avail_balance
De la cuenta
DONDE product_cd IN ('CHK', 'SAV', 'CD', 'MM');
```


re ldt W B kC

Página 10

Con el operador in , puede escribir una sola condición sin importar cuántas expresiones están en el set.

Usando subconsultas

Además de escribir su propio conjunto de expresiones, como ('CHK', 'SAV', 'CD', 'MM') , puede utilizar una subconsulta para generar un conjunto sobre la marcha. Por ejemplo, los cuatro productos los tipos utilizados en la consulta anterior tienen un product_type_cd de 'ACCOUNT' , así que ¿por qué no usar una subconsulta contra la tabla de productos para recuperar los cuatro códigos de producto en lugar de ex-nombrándolos explícitamente:

```
mysql> SELECT account_id, product_cd, cust_id, avail_balance
    -> DESDE cuenta
    -> DONDE product_cd IN (SELECCIONE product_cd FROM product
    -> DONDE product_type_cd = 'CUENTA');
+ ------------ + ------------ + --------- + ------------- - +
| account_id | product_cd | cust_id | avail_balance |
+ ------------ + ------------ + --------- + ------------- - +
| 3 | discos compactos | 1 | 3000,00 |
| 15 | discos compactos | 6 | 10000,00 |
| 17 | discos compactos | 7 | 5000,00 |
| 23 | discos compactos | 9 | 1500,00 |
| 1 | CHK | 1 | 1057.75 |
| 4 | CHK | 2 | 2258.02 |
| 7 | CHK | 3 | 1057.75 |
| 10 | CHK | 4 | 534.12 |
| 13 | CHK | 5 | 2237,97 |
| 14 | CHK | 6 | 122,37 |
| 18 | CHK | 8 | 3487.19 |
| 21 | CHK | 9 | 125,67 |
| 24 | CHK | 10 | 23575.12 |
| 28 | CHK | 12 | 38552.05 |
| 8 | MM | 3 | 2212.50 |
| 12 | MM | 4 | 5487.09 |
| 22 | MM | 9 | 9345.55 |
| 2 | SAV | 1 | 500,00 |
| 5 | SAV | 2 | 200,00 |
| 11 | SAV | 4 | 767,77 |
| 19 | SAV | 8 | 387,99 |
+ ------------ + ------------ + --------- + ------------- - +
21 filas en conjunto (0,11 seg)
```

La subconsulta devuelve un conjunto de cuatro valores y la consulta principal comprueba si el valor de la columna product_cd se puede encontrar en el conjunto que devolvió la subconsulta.

Usando no en

A veces desea ver si una expresión particular existe dentro de un conjunto de presiones y, a veces, desea ver si la expresión no existe. por
En estas situaciones, puede utilizar el operador not in :

re ldt W B kC

```
mysql> SELECT account_id, product_cd, cust_id, avail_balance
    -> DESDE cuenta
    -> DONDE product_cd NOT IN ('CHK', 'SAV', 'CD', 'MM');
+ ------------ + ------------ + --------- + ------------- - +
| account_id | product_cd | cust_id | avail_balance |
+ ------------ + ------------ + --------- + ------------- - +
|                 25 | AUTOBÚS              | 10 |                              0,00 |
|                 27 | AUTOBÚS              | 11 | 9345.55 |
|                 29 | SBL                  | 13 | 50000,00 |
+ ------------ + ------------ + --------- + ------------- - +
3 filas en conjunto (0.09 seg)
```

Esta consulta busca todas las cuentas que no son de cheques, ahorros, certificados de depósito o cuentas del mercado monetario.

Condiciones coincidentes

Hasta ahora, se le han presentado las condiciones que identifican una cadena exacta, un rango de cuerdas, o un conjunto de cuerdas; el tipo de condición final trata con coincidencias de cadenas parciales. Por ejemplo, es posible que desee buscar todos los empleados cuyo apellido comience con T. podría usar una función incorporada para quitar la primera letra de la columna lname , como en:

```
mysql> SELECT emp_id, fname, lname
    -> DESDE empleado
    -> DONDE IZQUIERDA (lname, 1) = 'T';
+ -------- + -------- + -------- +
| emp_id | fname | lname |
+ -------- + -------- + -------- +
| 3 | Robert | Tyler |
| 7 | Chris | Tucker |
| 18 | Rick | Tulman |
+ -------- + -------- + -------- +
3 filas en conjunto (0.01 seg)
```

Si bien la función incorporada left () hace el trabajo, no le brinda mucha flexibilidad.
En su lugar, puede utilizar caracteres comodín para crear expresiones de búsqueda, como se demuestra en la siguiente sección.

Usando comodines

Al buscar coincidencias de cadenas parciales, es posible que le interese:

- Cadenas que comienzan / terminan con un determinado carácter
- Cadenas que comienzan / terminan con una subcadena
- Cadenas que contienen cierto carácter en cualquier lugar dentro de la cadena
- Cadenas que contienen una subcadena en cualquier lugar dentro de la cadena
- Cadenas con un formato específico, independientemente de los caracteres individuales

Puede crear expresiones de búsqueda para identificar estas y muchas otras cadenas parciales coincide con el uso de los caracteres comodín que se muestran en la Tabla 4-4.

re ldt W B kC

Pagina 12

Tabla 4-4. Caracteres comodín

Coincidencias de caracteres comodín	
_	Exactamente un personaje
%	Cualquier número de caracteres (incluido 0)

El carácter de subrayado ocupa el lugar de un solo carácter, mientras que el signo de porcentaje puede tomar el lugar de un número variable de caracteres. Al construir condiciones que utiliza expresiones de búsqueda, utiliza el operador like , como en:

```
mysql> SELECT lname
    -> DESDE empleado
    -> DONDE lname COMO '_a% e%';
+ ----------- +
| lname |
+ ----------- +
| Barker |
| Hawthorne |
| Parker |
| Jameson |
+ ----------- +
4 filas en conjunto (0,00 seg)
```

La expresión de búsqueda en el ejemplo anterior especifica cadenas que contienen una a en el segunda posición y seguida de una e en cualquier otra posición de la cadena (incluida la Ultima posicion). La Tabla 4-5 muestra algunas expresiones de búsqueda más y sus interpretaciones.

Tabla 4-5. Expresiones de búsqueda de muestra

Expresión de búsqueda	Interpretación
F%	Cuerdas que comienzan con F
% t	Cadenas que terminan con t
% bas%	Cadenas que contienen la subcadena 'bas'
_ _t_	Cadenas de cuatro caracteres con una en la tercera posición
_ _ _-_ _-_ _ _ _	Cadenas de 11 caracteres con guiones en la cuarta y séptima posición

Puede utilizar el último ejemplo de la Tabla 4-5 para buscar clientes cuya identificación federal coincida el formato utilizado para los números de Seguro Social, como en:

```
mysql> SELECT cust_id, fed_id
    -> DEL cliente
    -> DONDE alimentado_id COMO '___-__-____';
+ --------- + ------------- +
| cust_id | fed_id |
+ --------- + ------------- +
| 1 | 111-11-1111 |
| 2 | 222-22-2222 |
| 3 | 333-33-3333 |
| 4 | 444-44-4444 |
| 5 | 555-55-5555 |
```


re ldt W B kC

Página 13

```
| 6 | 666-66-6666 |
| 7 | 777-77-7777 |
| 8 | 888-88-8888 |
| 9 | 999-99-9999 |
+ --------- + ------------- +
9 filas en conjunto (0.02 seg)
```

Los caracteres comodín funcionan bien para crear expresiones de búsqueda simples; si tus necesidades son un poco más sofisticados, sin embargo, puede utilizar varias expresiones de búsqueda, ya que demostrado por lo siguiente:

```
mysql> SELECT emp_id, fname, lname
    -> DESDE empleado
    -> DONDE lname COMO 'F%' O lname COMO 'G%';
+ -------- + ------- + ---------- +
| emp_id | fname | lname |
+ -------- + ------- + ---------- +
| 5 | John | Gooding |
| 6 | Helen | Fleming |
| 9 | Jane | Grossman |
| 17 | Beth | Fowler |
+ -------- + ------- + ---------- +
4 filas en conjunto (0,00 seg)
```

Esta consulta busca todos los empleados cuyo apellido comience con F o G.

Usando expresiones regulares

Si encuentra que los caracteres comodín no proporcionan suficiente flexibilidad, puede usar expresiones regulares para construir expresiones de búsqueda. Una expresión regular es, en esencia, una

expresión de búsqueda de esteroides. Si es nuevo en SQL, pero ha codificado usando programación lenguajes como Perl, entonces es posible que ya esté íntimamente familiarizado con los presiones. Si nunca ha utilizado expresiones regulares, puede consultar Dominio de expresiones regulares de Jeffrey EF Friedl (http://oreilly.com/catalog/9780596528126 /) (O'Reilly), ya que es un tema demasiado extenso para tratar de cubrirlo en este libro.

Esto es lo que la consulta anterior (busque todos los empleados cuyo apellido comience con F o G) se vería como usar la implementación de MySQL de expresiones regulares:

```
mysql> SELECT emp_id, fname, lname
    -> DESDE empleado
    -> DONDE lname REGEXP '^ [FG]';
+ -------- + ------- + ---------- +
| emp_id | fname | lname |
+ -------- + ------- + ---------- +
| 5 | John | Gooding |
| 6 | Helen | Fleming |
| 9 | Jane | Grossman |
| 17 | Beth | Fowler |
+ -------- + ------- + ---------- +
4 filas en conjunto (0,00 seg)
```

El operador regexp toma una expresión regular ('^ [FG]' en este ejemplo) y la aplica a la expresión en el lado izquierdo de la condición (la columna lname). La consulta

re ldt W B kC

Página 14

ahora contiene una sola condición usando una expresión regular en lugar de dos condiciones utilizando caracteres comodín.

Oracle Database y Microsoft SQL Server también admiten expresiones regulares. Con Oracle Database, usaría la función regexp_like en lugar del operador regexp mostrado en el ejemplo anterior, mientras que SQL Server permite que las expresiones regulares sean utilizado con el operador similar .

Nulo: Esa palabra de cuatro letras

Lo pospuse tanto como pude, pero es hora de abordar un tema que tiende a encontrarse con miedo, incertidumbre y pavor: el valor nulo . Nulo es la ausencia de un valor; antes de un empleado es despedido, por ejemplo, su columna end_date en la tabla de empleados debe ser nulo . No hay ningún valor que pueda asignarse a la columna end_date que haría

sentido en esta situación. Sin embargo, Null es un poco resbaladizo, ya que hay varios sabores de null :

No aplica
: Como la columna de identificación de empleado para una transacción que tuvo lugar en un cajero automático máquina

Valor aún no conocido
: Por ejemplo, cuando no se conoce la identificación federal en el momento en que se crea una fila de cliente

Valor indefinido
: Por ejemplo, cuando se crea una cuenta para un producto que aún no se ha agregado a la base de datos

> Algunos teóricos argumentan que debería haber una expresión diferente para cubrir cada una de estas (y más) situaciones, pero la mayoría de los profesionales estarían de acuerdo que tener múltiples valores nulos sería demasiado confuso.

Cuando trabaje con null , debe recordar:

- Una expresión puede ser nula, pero nunca puede ser igual a nula.
- Dos nulos nunca son iguales entre sí.

Para probar si una expresión es nula , debe usar el operador es nulo , como se demuestra onstrated por lo siguiente:

```
mysql> SELECT emp_id, fname, lname, superior_emp_id
    -> DESDE empleado
    -> DONDE superior_emp_id ES NULO;
+ -------- + --------- + ------- + ----------------- +
| emp_id | fname | lname | superior_emp_id |
+ -------- + --------- + ------- + ----------------- +
| 1 | Michael | Smith | NULL |
+ -------- + --------- + ------- + ----------------- +
1 fila en conjunto (0,00 seg)
```


re ldt W B kC

Página 15

Esta consulta devuelve todos los empleados que no tienen un jefe (¿no sería bueno?).
Aquí está la misma consulta usando = null en lugar de es null :

```
mysql> SELECT emp_id, fname, lname, superior_emp_id
```

```
    -> DESDE empleado
    -> DONDE superior_emp_id = NULL;
Conjunto vacío (0.01 seg)
```

Como puede ver, la consulta analiza y ejecuta, pero no devuelve ninguna fila. Esto es un error común cometido por programadores de SQL sin experiencia y el servidor de base de datos no le alertará sobre su error, así que tenga cuidado al construir condiciones que prueben nulo .

Si usted quiere ver si un valor ha sido asignado a una columna, puede utilizar la es no operador nulo , como en:

```
mysql> SELECT emp_id, fname, lname, superior_emp_id
    -> DESDE empleado
    -> DONDE superior_emp_id NO ES NULO;
+ -------- + ---------- + ----------- + ----------------- +
| emp_id | fname | lname | superior_emp_id |
+ -------- + ---------- + ----------- + ----------------- +
| 2 | Susan | Barker | 1 |
| 3 | Robert | Tyler | 1 |
| 4 | Susan | Hawthorne | 3 |
| 5 | John | Gooding | 4 |
| 6 | Helen | Fleming | 4 |
| 7 | Chris | Tucker | 6 |
| 8 | Sarah | Parker | 6 |
| 9 | Jane | Grossman | 6 |
| 10 | Paula | Roberts | 4 |
| 11 | Thomas | Ziegler | 10 |
| 12 | Samantha | Jameson | 10 |
| 13 | John | Blake | 4 |
| 14 | Cindy | Mason | 13 |
| 15 | Frank | Portman | 13 |
| 16 | Theresa | Markham | 4 |
| 17 | Beth | Fowler | 16 |
| 18 | Rick | Tulman | 16 |
+ -------- + ---------- + ----------- + ----------------- +
17 filas en conjunto (0,00 seg)
```

Esta versión de la consulta devuelve los otros 17 empleados que, a diferencia de Michael Smith, tener un jefe.

Antes de dejar el nulo a un lado por un tiempo, sería útil investigar un punto más. trampa potencial. Suponga que se le ha pedido que identifique a todos los empleados que no administrado por Helen Fleming (cuya identificación de empleado es 6). Tu primer instinto podría ser para hacer lo siguiente:

```
mysql> SELECT emp_id, fname, lname, superior_emp_id
    -> DESDE empleado
    -> DONDE superior_emp_id! = 6;
```


re ldt W B kC

Página 16

```
+ -------- + ---------- + ----------- + ----------------- +
| emp_id | fname | lname | superior_emp_id |
+ -------- + ---------- + ----------- + ----------------- +
| 2 | Susan | Barker | 1 |
| 3 | Robert | Tyler | 1 |
| 4 | Susan | Hawthorne | 3 |
| 5 | John | Gooding | 4 |
| 6 | Helen | Fleming | 4 |
| 10 | Paula | Roberts | 4 |
| 11 | Thomas | Ziegler | 10 |
| 12 | Samantha | Jameson | 10 |
| 13 | John | Blake | 4 |
| 14 | Cindy | Mason | 13 |
| 15 | Frank | Portman | 13 |
| 16 | Theresa | Markham | 4 |
| 17 | Beth | Fowler | 16 |
| 18 | Rick | Tulman | 16 |
+ -------- + ---------- + ----------- + ----------------- +
14 filas en conjunto (0,00 seg)
```

Si bien es cierto que estos 14 empleados no trabajan para Helen Fleming, si miras cuidadosamente en los datos, verá que hay un empleado más que no trabaja para Helen, que no figura aquí. Ese empleado es Michael Smith, y su

La columna superior_emp_id es nula (porque él es el gran queso). Para responder la pregunta correctamente, por lo tanto, debe tener en cuenta la posibilidad de que algunas filas mantener un nulo en la columna superior_emp_id :

```
mysql> SELECT emp_id, fname, lname, superior_emp_id
    -> DESDE empleado
    -> DONDE superior_emp_id! = 6 O superior_emp_id ES NULO;
+ -------- + ---------- + ----------- + ----------------- +
| emp_id | fname | lname | superior_emp_id |
+ -------- + ---------- + ----------- + ----------------- +
| 1 | Michael | Smith | NULL |
| 2 | Susan | Barker | 1 |
| 3 | Robert | Tyler | 1 |
| 4 | Susan | Hawthorne | 3 |
| 5 | John | Gooding | 4 |
| 6 | Helen | Fleming | 4 |
| 10 | Paula | Roberts | 4 |
| 11 | Thomas | Ziegler | 10 |
| 12 | Samantha | Jameson | 10 |
| 13 | John | Blake | 4 |
| 14 | Cindy | Mason | 13 |
| 15 | Frank | Portman | 13 |
| 16 | Theresa | Markham | 4 |
| 17 | Beth | Fowler | 16 |
| 18 | Rick | Tulman | 16 |
+ -------- + ---------- + ----------- + ----------------- +
15 filas en conjunto (0,00 seg)
```

El conjunto de resultados ahora incluye a los 15 empleados que no trabajan para Helen. Al trabajar con una base de datos con la que no está familiarizado, es una buena idea averiguar qué columnas

re ldt W B kC

Página 17

en una tabla, permita nulos para que pueda tomar las medidas adecuadas con su filtro diciones para evitar que los datos se escapen.

Prueba tus conocimientos

Los siguientes ejercicios ponen a prueba su comprensión de las condiciones del filtro. Consulte el Apéndice dix C para soluciones.

Los siguientes datos de transacciones se utilizan para los dos primeros ejercicios:

Txn_id	Txn_date	Account_id	Txn_type_cd	Importe
1	2005-02-22	101	CDT	1000,00
2	2005-02-23	102	CDT	525,75
3	2005-02-24	101	DBT	100,00
4	2005-02-24	103	CDT	55
5	2005-02-25	101	DBT	50
6	2005-02-25	103	DBT	25
7	2005-02-25	102	CDT	125,37
8	2005-02-26	103	DBT	10
9	2005-02-27	101	CDT	75

Ejercicio 4-1

¿Cuáles de los ID de transacción se devolverían con las siguientes condiciones de filtro?

```
txn_date <'2005-02-26' Y (txn_type_cd = 'DBT' O cantidad> 100)
```

Ejercicio 4-2

¿Cuáles de los ID de transacción se devolverían con las siguientes condiciones de filtro?

```
account_id IN (101,103) Y NO (txn_type_cd = 'DBT' O cantidad> 100)
```

Ejercicio 4-3

Construya una consulta que recupere todas las cuentas abiertas en 2002.

Ejercicio 4-4

Construya una consulta que encuentre todos los clientes no comerciales cuyo apellido contenga un a en la segunda posición y una e en cualquier lugar después de a.

re ldt W B kC

Página 18

re ldt W B kC