CAPÍTULO 8

Agrupación y agregados

Los datos generalmente se almacenan en el nivel más bajo de granularidad que necesita cualquiera de las bases de datos. usuarios; si Chuck en contabilidad necesita mirar las transacciones de clientes individuales, entonces es necesario que haya una tabla en la base de datos que almacene transacciones individuales. Eso no Sin embargo, significa que todos los usuarios deben tratar los datos tal como están almacenados en la base de datos. El enfoque de este capítulo es cómo se pueden agrupar y agregar los datos para permitir a los usuarios para interactuar con él en un nivel más alto de granularidad que lo que está almacenado en la base de datos.

Conceptos de agrupación

A veces querrá encontrar tendencias en sus datos que requerirán el servidor de base de datos cocinar los datos un poco antes de que pueda generar los resultados que está buscando. por Por ejemplo, digamos que está a cargo de las operaciones en el banco y le gustaría para saber cuántas cuentas abre cada cajero de banco. Podrías emitir una consulta simple para mirar los datos sin procesar:

```
mysql> SELECCIONAR open_emp_id
    -> DESDE cuenta;
+ ------------- +
| open_emp_id |
+ ------------- +
| 1 |
| 1 |
| 1 |
| 1 |
| 1 |
| 1 |
```

```
|                  1 |
|                  1 |
|                 10 |
|                 10 |
|                 10 |
|                 10 |
|                 10 |
|                 10 |
|                 10 |
|                 13 |
```


re ldt W B kC

Página 2

```
|                 13 |
|                 13 |
|                 16 |
|                 16 |
|                 16 |
|                 16 |
|                 16 |
|                 16 |
+ ------------- +
24 filas en conjunto (0.01 seg)
```

Con solo 24 filas en la tabla de cuentas , es relativamente fácil ver que cuatro los empleados abrieron cuentas y el ID de empleado 16 ha abierto seis cuentas; cómo- nunca, si el banco tiene docenas de empleados y miles de cuentas, este enfoque resultaría tedioso y propenso a errores.

En su lugar, puede pedirle al servidor de la base de datos que agrupe los datos por usted utilizando el grupo por cláusula. Esta es la misma consulta pero empleando una cláusula group by para agrupar la cuenta datos por ID de empleado:

```
mysql> SELECCIONAR open_emp_id
    -> DESDE cuenta
    -> GRUPO POR open_emp_id;
+ ------------- +
| open_emp_id |
+ ------------- +
|                  1 |
|                 10 |
|                 13 |
|                 16 |
+ ------------- +
```

```
4 filas en conjunto (0,00 seg)
```

El conjunto de resultados contiene una fila para cada valor distinto en la columna open_emp_id , resultando en cuatro filas en lugar de las 24 filas completas. El motivo del conjunto de resultados más pequeño es que cada uno de los cuatro empleados abrió más de una cuenta. Para ver cuantos cuentas que abrió cada cajero, puede usar una función agregada en la cláusula de selección para cuente el número de filas en cada grupo:

```
mysql> SELECT open_emp_id, COUNT (*) how_many
    -> DESDE cuenta
    -> GRUPO POR open_emp_id;
+ ------------- + ---------- +
| open_emp_id | how_many |
+ ------------- + ---------- +
|                1 |                8 |
|               10 |                7 |
|               13 |                3 |
|               16 |                6 |
+ ------------- + ---------- +
4 filas en conjunto (0,00 seg)
```

La función agregada count () cuenta el número de filas en cada grupo, y la El asterisco le dice al servidor que cuente todo en el grupo. Usando la combinación de un

re ldt W B kC

Página 3

group by cláusula y la función agregada count () , puede generar exactamente los datos necesarios para responder a la pregunta comercial sin tener que mirar los datos sin procesar.

Al agrupar datos, es posible que deba filtrar los datos no deseados de su conjunto de resultados basado en grupos de datos en lugar de basarse en los datos brutos. Dado que la cláusula group by se ejecuta después de que se haya evaluado la cláusula where , no puede agregar condiciones de filtro a su where cláusula para este propósito. Por ejemplo, aquí hay un intento de filtrar cualquier caso. donde un empleado ha abierto menos de cinco cuentas:

```
mysql> SELECT open_emp_id, COUNT (*) how_many
    -> DESDE cuenta
    -> DONDE CONTAR (*)> 4
    -> GRUPO POR open_emp_id;
ERROR 1111 (HY000): uso no válido de la función de grupo
```

No puede hacer referencia al recuento de funciones agregadas (*) en su cláusula where , porque el los grupos aún no se han generado en el momento en que se evalúa la cláusula where . En lugar,

debe poner sus condiciones de filtro de grupo en la cláusula de tener . Esto es lo que hace la consulta se vería como usar tener :

```
mysql> SELECT open_emp_id, COUNT (*) how_many
    -> DESDE cuenta
    -> GRUPO POR open_emp_id
    -> TENIENDO CUENTA (*)> 4;
+ ------------- + ---------- +
| open_emp_id | how_many |
+ ------------- + ---------- +
|                1 |                8 |
|               10 |                7 |
|               16 |                6 |
+ ------------- + ---------- +
3 filas en conjunto (0,00 seg)
```

Debido a que los grupos que contienen menos de cinco miembros se han filtrado a través de la cláusula de tener , el conjunto de resultados ahora contiene solo aquellos empleados que han abierto cinco o más cuentas, eliminando así el ID de empleado 13 de los resultados.

Funciones agregadas

Las funciones agregadas realizan una operación específica en todas las filas de un grupo. A pesar de que cada servidor de base de datos tiene su propio conjunto de funciones agregadas especializadas, el común Las funciones agregadas implementadas por todos los servidores principales incluyen:

Max ()
: Devuelve el valor máximo dentro de un conjunto

Min ()
: Devuelve el valor mínimo dentro de un conjunto

Promedio ()
: Devuelve el valor medio de un conjunto.

re ldt W B kC

Página 4

Suma()
: Devuelve la suma de los valores de un conjunto.

Contar()

Devuelve el número de valores de un conjunto.

A continuación, se muestra una consulta que utiliza todas las funciones agregadas comunes para analizar las saldos de todas las cuentas corrientes:

```
mysql> SELECT MAX (avail_balance) max_balance,
    -> MIN (disponible_balance) min_balance,
    -> AVG (avail_balance) avg_balance,
    -> SUM (disponible_balance) tot_balance,
    -> COUNT (*) núm_cuentas
    -> DESDE cuenta
    -> DONDE product_cd = 'CHK';
+ ------------ + ------------ + ------------ + ------- ------ + -------------- +
| max_balance | min_balance | avg_balance | tot_balance | num_accounts |
+ ------------ + ------------ + ------------ + ------- ------ + -------------- +
| 38552.05 | 122,37 | 7300.800985 | 73008.01 | 10 |
+ ------------ + ------------ + ------------ + ------- ------ + -------------- +
1 fila en conjunto (0.09 seg)
```

Los resultados de esta consulta le indican que, en las 10 cuentas corrientes del tabla de cuenta , hay un saldo máximo de $ 38,552.05, un saldo mínimo de $ 122.37, un saldo promedio de $ 7,300.80 y un saldo total en las 10 cuentas de $ 73,008.01. Con suerte, esto le dará una apreciación del papel de estos agregados funciones; las siguientes subsecciones aclaran aún más cómo puede utilizar estas funciones.

Grupos implícitos versus explícitos

En el ejemplo anterior, cada valor devuelto por la consulta es generado por un agregado función, y las funciones agregadas se aplican a través del grupo de filas especificado por la condición de filtro product_cd = 'CHK' . Dado que no hay una cláusula group by , hay una grupo implícito único (todas las filas devueltas por la consulta).

En la mayoría de los casos, sin embargo, querrá recuperar columnas adicionales junto con umns generados por funciones agregadas. ¿Y si, por ejemplo, quisiera ampliar la consulta anterior para ejecutar las mismas cinco funciones agregadas para cada tipo de producto, en lugar de solo para cuentas corrientes? Para esta consulta, querrá recuperar el columna product_cd junto con las cinco funciones agregadas, como en:

```
SELECCIONAR product_cd,
    MAX (disponible_balance) max_balance,
    MIN (disponible_balance) min_balance,
    AVG (avail_balance) avg_balance,
    SUM (disponible_balance) tot_balance,
    COUNT (*) núm_cuentas
De la cuenta;
```

Sin embargo, si intenta ejecutar la consulta, recibirá el siguiente error:

re ldt W B kC

Página 5

```
ERROR 1140 (42000): Mezcla de columnas GROUP (MIN (), MAX (), COUNT (), ...) sin GROUP
columnas es ilegal si no hay una cláusula GROUP BY
```

Si bien puede resultarle obvio que desea que las funciones agregadas se apliquen a cada conjunto de productos encontrados en la tabla de cuentas , esta consulta falla porque no ha especificó explícitamente cómo se deben agrupar los datos. Por lo tanto, deberá agregar un Cláusula group by para especificar sobre qué grupo de filas deben ser las funciones agregadas aplicado:

```
mysql> SELECT product_cd,
    -> MAX (disponible_balance) max_balance,
    -> MIN (disponible_balance) min_balance,
    -> AVG (avail_balance) avg_balance,
    -> SUM (disponible_balance) tot_balance,
    -> CONTAR (*) num_accts
    -> DESDE cuenta
    -> GRUPO POR producto_cd;
+ ------------ + ------------- + ------------- + -------- ------ + ------------- + ----------- +
| product_cd | max_balance | min_balance | avg_balance | tot_balance | num_accts |
+ ------------ + ------------- + ------------- + -------- ------ + ------------- + ----------- +
| AUTOBÚS | 9345.55 | 0,00 | 4672.774902 | 9345.55 | 2 |
| discos compactos | 10000,00 | 1500,00 | 4875.000000 | 19500.00 | 4 |
| CHK | 38552.05 | 122,37 | 7300.800985 | 73008.01 | 10 |
| MM | 9345.55 | 2212.50 | 5681.713216 | 17045.14 | 3 |
| SAV | 767,77 | 200,00 | 463,940002 | 1855,76 | 4 |
| SBL | 50000,00 | 50000,00 | 50000.000000 | 50000,00 | 1 |
+ ------------ + ------------- + ------------- + -------- ------ + ------------- + ----------- +
6 filas en conjunto (0,00 seg)
```

Con la inclusión de la cláusula group by , el servidor sabe agrupar filas tener el mismo valor en la columna product_cd primero y luego aplicar los cinco agregados funciones para cada uno de los seis grupos.

Contando valores distintos

Al usar la función count () para determinar el número de miembros en cada grupo, tiene la opción de contar a todos los miembros del grupo, o contar solo los valores distintos para una columna en todos los miembros del grupo. Por ejemplo, considere el los siguientes datos, que muestran el empleado responsable de abrir cada cuenta:

```
mysql> SELECT account_id, open_emp_id
    -> DESDE cuenta
    -> ORDEN POR open_emp_id;
+ ------------ + ------------- +
| account_id | open_emp_id |
+ ------------ + ------------- +
| 8 | 1 |
| 9 | 1 |
| 10 | 1 |
| 12 | 1 |
| 13 | 1 |
| 17 | 1 |
| 18 | 1 |
```


re ldt W B kC

Página 6

```
|                 19 |                  1 |
|                  1 |                 10 |
|                  2 |                 10 |
|                  3 |                 10 |
|                  4 |                 10 |
|                  5 |                 10 |
|                 14 |                 10 |
|                 22 |                 10 |
|                  6 |                 13 |
|                  7 |                 13 |
|                 24 |                 13 |
|                 11 |                 16 |
|                 15 |                 16 |
|                 16 |                 16 |
|                 20 |                 16 |
|                 21 |                 16 |
|                 23 |                 16 |
+ ------------ + ------------- +
24 filas en conjunto (0,00 seg)
```

Como puede ver, cuatro empleados diferentes abrieron varias cuentas (empleado ID 1, 10, 13 y 16). Digamos que, en lugar de realizar un recuento manual, desea para crear una consulta que cuente el número de empleados que han abierto cuentas. Si aplica la función count () a la columna open_emp_id , verá lo siguiente resultados:

```
mysql> SELECT COUNT (open_emp_id)
    -> DESDE cuenta;
+ ------------------- +
| COUNT (open_emp_id) |
+ ------------------- +
|                  24 |
+ ------------------- +
1 fila en conjunto (0,00 seg)
```

En este caso, especificar la columna open_emp_id como la columna a contar genera los mismos resultados que especificar el recuento (*) . Si desea contar valores distintos en el grupo en lugar de simplemente contar el número de filas del grupo, debe especificar el palabra clave distinta , como en:

```
mysql> SELECT COUNT (DISTINCT open_emp_id)
    -> DESDE cuenta;
+ ---------------------------- +
| COUNT (DISTINCT open_emp_id) |
+ ---------------------------- +
|                            4 |
+ ---------------------------- +
1 fila en conjunto (0,00 seg)
```

Al especificar distinto , por lo tanto, la función count () examina los valores de una columna para cada miembro del grupo con el fin de encontrar y eliminar duplicados, en lugar de simplemente contando el número de valores en el grupo.

re ldt W B kC

Página 7

Usando Expresiones

Además de usar columnas como argumentos para agregar funciones, puede construir expresiones siones para usar como argumentos. Por ejemplo, es posible que desee encontrar el valor máximo de depósitos pendientes en todas las cuentas, que se calcula restando la disponibilidad saldo del saldo pendiente. Puede lograr esto a través de la siguiente consulta:

```
mysql> SELECT MAX (pendiente_balance - avail_balance) max_uncleared
    -> DESDE cuenta;
+ --------------- +
| max_uncleared |
+ --------------- +
| 660,00 |
+ --------------- +
1 fila en conjunto (0,00 seg)
```

Si bien este ejemplo utiliza una expresión bastante simple, las expresiones utilizadas como argumentos para Las funciones agregadas pueden ser tan complejas como sea necesario, siempre que devuelvan un número, cadena o fecha. En el Capítulo 11, le muestro cómo puede usar expresiones de casos con funciones agregadas para determinar si una fila en particular debe o no debe ser incluido en una agregación.

Cómo se manejan los nulos

Al realizar agregaciones o, de hecho, cualquier tipo de cálculo numérico, debe Siempre considere cómo los valores nulos pueden afectar el resultado de su cálculo. A ilustrar, construiré una tabla simple para contener datos numéricos y completarla con el conjunto {1, 3, 5}:

```
mysql> CREAR TABLA number_tbl
    -> (val SMALLINT);
Consulta OK, 0 filas afectadas (0.01 seg)
```

```
mysql> INSERT INTO number_tbl VALUES (1);
Consulta correcta, 1 fila afectada (0,00 segundos)

mysql> INSERT INTO number_tbl VALUES (3);
Consulta correcta, 1 fila afectada (0,00 segundos)

mysql> INSERT INTO number_tbl VALUES (5);
Consulta correcta, 1 fila afectada (0,00 segundos)
```

Considere la siguiente consulta, que realiza cinco funciones agregadas en el conjunto de números:

```
mysql> SELECT COUNT (*) num_rows,
    -> COUNT (val) num_vals,
    -> SUM (val) total,
    -> MAX (val) max_val,
    -> AVG (val) avg_val
    -> FROM number_tbl;
+ ---------- + ---------- + ------- + --------- + --------- +
```


re ldt W B kC

Página 8

```
| num_rows | num_vals | total | max_val | avg_val |
+ ---------- + ---------- + ------- + --------- + --------- +
| 3 | 3 | 9 | 5 | 3.0000 |
+ ---------- + ---------- + ------- + --------- + --------- +
1 fila en conjunto (0.08 seg)
```

Los resultados son los esperados: tanto count (*) como count (val) devuelven el valor 3 , sum (val) devuelve el valor 9 , max (val) devuelve 5 y avg (val) devuelve 3 . A continuación, lo haré agregue un valor nulo a la tabla number_tbl y ejecute la consulta nuevamente:

```
mysql> INSERT INTO number_tbl VALUES (NULL);
Consulta correcta, 1 fila afectada (0,01 s)

mysql> SELECT COUNT (*) num_rows,
    -> COUNT (val) num_vals,
    -> SUM (val) total,
    -> MAX (val) max_val,
    -> AVG (val) avg_val
    -> FROM number_tbl;
+ ---------- + ---------- + ------- + --------- + --------- +
| num_rows | num_vals | total | max_val | avg_val |
+ ---------- + ---------- + ------- + --------- + --------- +
| 4 | 3 | 9 | 5 | 3.0000 |
```

```
+ ---------- + ---------- + ------- + --------- + --------- +
1 fila en conjunto (0,00 seg)
```

Incluso con la adición del valor nulo a la tabla, las funciones sum () , max () y avg ()
todas devuelven los mismos valores, lo que indica que ignoran cualquier valor nulo encontrado.
La función count (*) ahora devuelve el valor 4 , que es válido ya que la tabla number_tbl
contiene cuatro filas, mientras que la función count (val) aún devuelve el valor 3 . La diferencia
es que count (*) cuenta el número de filas, mientras que count (val) cuenta el número
ber de los valores contenidos en la columna val e ignora los valores nulos encontrados.

Generación de grupos

La gente rara vez está interesada en ver datos sin procesar; en cambio, las personas que participan en los datos
El análisis querrá manipular los datos sin procesar para adaptarse mejor a sus necesidades. Ejemplos de
Las manipulaciones de datos comunes incluyen:

- Generar totales para una región geográfica, como el total de ventas europeas
- Encontrar valores atípicos, como el principal vendedor de 2005
- Determinar frecuencias, como el número de cuentas nuevas abiertas para cada
  rama

Para responder a este tipo de consultas, deberá solicitar al servidor de la base de datos que agrupe las filas
juntos por una o más columnas o expresiones. Como ya has visto en varios
Por ejemplo, la cláusula group by es el mecanismo para agrupar datos dentro de una consulta. En
En esta sección, verá cómo agrupar datos por una o más columnas, cómo agrupar
datos mediante expresiones y cómo generar resúmenes dentro de grupos.

re ldt W B kC

Página 9

Agrupación de una sola columna

Los grupos de una sola columna son el tipo de agrupación más simple y más utilizado. Si tu
desea encontrar los saldos totales para cada producto, por ejemplo, solo necesita agrupar en
la columna account.product_cd , como en:

```
mysql> SELECT product_cd, SUM (avail_balance) prod_balance
    -> DESDE cuenta
```

```
        -> GRUPO POR producto_cd;
+ ------------ + -------------- +
| product_cd | prod_balance |
+ ------------ + -------------- +
| AUTOBÚS | 9345.55 |
| discos compactos | 19500.00 |
| CHK | 73008.01 |
| MM | 17045.14 |
| SAV | 1855,76 |
| SBL | 50000,00 |
+ ------------ + -------------- +
6 filas en conjunto (0,00 seg)
```

Esta consulta genera seis grupos, uno para cada producto, y luego suma los disponibles saldos para cada miembro del grupo.

Agrupación multicolumna

En algunos casos, es posible que desee generar grupos que abarquen más de una columna. Ampliando el ejemplo anterior, imagina que quieres encontrar los saldos totales no solo para cada producto, sino para los productos y las ramas (p. ej., cuál es el total saldo de todas las cuentas corrientes abiertas en la sucursal de Woburn?). El siguiente ex- ample muestra cómo puede lograr esto:

```
mysql> SELECT product_cd, open_branch_id,
        -> SUM (disponible_balance) tot_balance
        -> DESDE cuenta
        -> GROUP BY product_cd, open_branch_id;
+ ------------ + ---------------- + ------------- +
| product_cd | open_branch_id | tot_balance |
+ ------------ + ---------------- + ------------- +
| AUTOBÚS | 2 | 9345.55 |
| AUTOBÚS | 4 | 0,00 |
| discos compactos | 1 | 11500,00 |
| discos compactos | 2 | 8000,00 |
| CHK | 1 | 782,16 |
| CHK | 2 | 3315.77 |
| CHK | 3 | 1057.75 |
| CHK | 4 | 67852,33 |
| MM | 1 | 14832.64 |
| MM | 3 | 2212.50 |
| SAV | 1 | 767,77 |
| SAV | 2 | 700,00 |
| SAV | 4 | 387,99 |
| SBL | 3 | 50000,00 |
```


re ldt W B kC

Página 10

```
+ ------------ + ----------------- + -------------- +
14 filas en conjunto (0,00 seg)
```

Esta versión de la consulta genera 14 grupos, uno por cada combinación de producto y sucursal que se encuentran en la tabla de cuentas . Junto con agregar la columna open_branch_id a la cláusula select , también lo agregué a la cláusula group by , ya que open_branch_id es recuperado de una tabla y no se genera a través de una función agregada.

Agrupación a través de expresiones

Además de usar columnas para agrupar datos, puede crear grupos basados en los valores generado por expresiones. Considere la siguiente consulta, que agrupa a los empleados por el año en que empezaron a trabajar para el banco:

```
mysql> SELECT EXTRACT (YEAR FROM start_date) año,
    -> COUNT (*) cuántos
    -> DESDE empleado
    -> GRUPO POR EXTRACTO (AÑO DESDE fecha_inicio);
+ ------ + ---------- +
| año | how_many |
+ ------ + ---------- +
| 2004 | 2 |
| 2005 | 3 |
| 2006 | 8 |
| 2007 | 3 |
| 2008 | 2 |
+ ------ + ---------- +
5 filas en conjunto (0,15 seg)
```

Esta consulta emplea una expresión bastante simple, que usa la función extract () para devuelve solo la parte del año de una fecha, para agrupar las filas en la tabla de empleados .

Generando acumulaciones

En "Agrupación de varias columnas" en la página 151, mostré un ejemplo que genera saldos de cuentas para cada producto y rama. Digamos, sin embargo, que junto con el saldos totales para cada combinación de producto / sucursal, también desea saldos totales para cada producto distinto. Puede ejecutar una consulta adicional y fusionar los resultados, podría cargar los resultados de la consulta en una hoja de cálculo, o podría crear un script en Perl, Programa Java, o algún otro mecanismo para tomar esos datos y realizar los cálculos. Mejor aún, puede usar la opción con rollup para tener la base de datos servidor hace el trabajo por usted. Aquí está la consulta revisada usando con rollup en el grupo por cláusula:

```
mysql> SELECT product_cd, open_branch_id,
    -> SUM (disponible_balance) tot_balance
    -> DESDE cuenta
    -> GROUP BY product_cd, open_branch_id CON ROLLUP;

+ ------------ + ----------------- + -------------- +
| product_cd | open_branch_id | tot_balance |
```


re ldt W B kC

Página 11

```
+ ------------ + ---------------- + ------------- +
| AUTOBÚS      |                2 | 9345.55 |
| AUTOBÚS      |                4 |                0,00 |
| AUTOBÚS      |             NULL | 9345.55 |
| discos compactos |             1 | 11500,00 |
| discos compactos |             2 | 8000,00 |
| discos compactos |          NULL | 19500.00 |
| CHK          |                1 | 782,16 |
| CHK          |                2 | 3315.77 |
| CHK          |                3 | 1057.75 |
| CHK          |                4 | 67852,33 |
| CHK          |             NULL | 73008.01 |
| MM           |                1 | 14832.64 |
| MM           |                3 | 2212.50 |
| MM           |             NULL | 17045.14 |
| SAV          |                1 | 767,77 |
| SAV          |                2 | 700,00 |
| SAV          |                4 | 387,99 |
| SAV          |             NULL | 1855,76 |
| SBL          |                3 | 50000,00 |
| SBL          |             NULL | 50000,00 |
| NULL         |             NULL | 170754.46 |
+ ------------ + ---------------- + ------------- +
21 filas en conjunto (0.02 seg)
```

Ahora hay siete filas adicionales en el conjunto de resultados, una para cada uno de los seis distintos productos y uno para el total general (todos los productos combinados). Para los seis productos rollups, se proporciona un valor nulo para la columna open_branch_id , ya que el rollup se realizado en todas las ramas. Mirando la tercera línea de la salida, por ejemplo, verá que se depositó un total de $ 9,345.55 en las cuentas de BUS en todas las sucursales. Para la fila de gran total, se proporciona un valor nulo para product_cd y columnas open_branch_id ; la última línea de salida muestra un total de $ 170,754.46 en todos productos y ramas.

> Si está utilizando Oracle Database, debe utilizar una sintaxis para indicar que desea realizar un resumen. El grupo por cláusula para la consulta anterior se vería de la siguiente manera cuando se usa Oracle:
>
> ```
> GRUPO POR ROLLUP (product_cd, open_branch_id)
> ```
>
> La ventaja de esta sintaxis es que le permite realizar acumulaciones en un subconjunto de las columnas de la cláusula group by . Si está agrupando por columnas a, byc, por ejemplo, podría indicar que el servidor debe realizar acumulaciones solo en byc a través de lo siguiente:
>
> ```
> AGRUPAR POR a, ROLLUP (b, c)
> ```

Si, junto con los totales por producto, también desea calcular los totales por rama, entonces puede usar la opción with cube , que genera filas de resumen para todas las combinaciones posibles naciones de las columnas de agrupación. Desafortunadamente, con cube no está disponible en la versión 6.0 de MySQL, pero está disponible con SQL Server y Oracle Database. Aquí hay un

re ldt W B kC

Pagina 12

ejemplo usando con cube , pero he eliminado el indicador mysql> para mostrar que la consulta aún no se puede realizar con MySQL:

```
SELECT product_cd, open_branch_id,
    SUM (disponible_balance) tot_balance
De la cuenta
GROUP BY product_cd, open_branch_id WITH CUBE ;
+ ------------ + ---------------- + ------------- +
| product_cd | open_branch_id | tot_balance |
+ ------------ + ---------------- + ------------- +
| NULL |                    NULL | 170754.46 |
| NULL |                       1 | 27882.57 |
| NULL |                       2 | 21361.32 |
| NULL |                       3 | 53270.25 |
| NULL |                       4 | 68240.32 |
| AUTOBÚS |                     2 | 9345.55 |
| AUTOBÚS |                     4 |                    0,00 |
| AUTOBÚS |                  NULL | 9345.55 |
| discos compactos |            1 | 11500,00 |
| discos compactos |            2 | 8000,00 |
| discos compactos |         NULL | 19500.00 |
| CHK |                           1 | 782,16 |
| CHK |                           2 | 3315.77 |
| CHK |                           3 | 1057.75 |
| CHK |                           4 | 67852,33 |
| CHK |                        NULL | 73008.01 |
| MM |                            1 | 14832.64 |
| MM |                            3 | 2212.50 |
| MM |                         NULL | 17045.14 |
| SAV |                           1 | 767,77 |
| SAV |                           2 | 700,00 |
| SAV |                           4 | 387,99 |
| SAV |                        NULL | 1855,76 |
| SBL |                           3 | 50000,00 |
| SBL |                        NULL | 50000,00 |
+ ------------ + ---------------- + ------------- +
25 filas en conjunto (0.02 seg)
```

El uso con cubo genera cuatro filas más que con la versión acumulada de la consulta, uno para cada uno de los cuatro ID de sucursal. Al igual que con el rollup , los valores nulos se colocan en

la columna product_cd para indicar que se está realizando un resumen de rama.

Una vez más, si está utilizando Oracle Database, debe utilizar un ligero sintaxis diferente para indicar que desea realizar una operación de cubo. La cláusula group by para la consulta anterior tendría el siguiente aspecto cuando usando Oracle:

```
GRUPO POR CUBO (product_cd, open_branch_id)
```


re ldt W B kC

Página 13

Condiciones de filtro de grupo

En el Capítulo 4, le presenté varios tipos de condiciones de filtro y mostré cómo puede usarlos en la cláusula where . Al agrupar datos, también puede aplicar filtros condiciones a los datos después de que se hayan generado los grupos. La cláusula de tener es donde debe colocar estos tipos de condiciones de filtro. Considere el siguiente ejemplo:

```
mysql> SELECT product_cd, SUM (avail_balance) prod_balance
    -> DESDE cuenta
    -> DONDE estado = 'ACTIVO'
    -> GRUPO POR product_cd
    -> TENIENDO SUMA (balance_disponible)> = 10000;
+ ------------ + -------------- +
| product_cd | prod_balance |
+ ------------ + -------------- +
| discos compactos | 19500.00 |
| CHK | 73008.01 |
| MM | 17045.14 |
| SBL | 50000,00 |
+ ------------ + -------------- +
4 filas en conjunto (0,00 seg)
```

Esta consulta tiene dos condiciones de filtro: una en la cláusula where , que filtra inactivo cuentas, y el otro en la cláusula de tener , que filtra cualquier producto cuyo total el saldo disponible es menos de $ 10,000. Por tanto, uno de los filtros actúa sobre los datos antes de

agrupados, y el otro filtro actúa sobre los datos después de que se han creado los grupos. Si tu Si coloca ambos filtros por error en la cláusula where , verá el siguiente error:

```
mysql> SELECT product_cd, SUM (avail_balance) prod_balance
    -> DESDE cuenta
    -> DONDE estado = 'ACTIVO'
    -> Y SUMA (balance_disponible)> 10000
    -> GRUPO POR producto_cd;
ERROR 1111 (HY000): uso no válido de la función de grupo
```

Esta consulta falla porque no puede incluir una función agregada en el dónde de una consulta cláusula. Esto se debe a que los filtros en la cláusula where se evalúan antes de la agrupación ocurre, por lo que el servidor aún no puede realizar ninguna función en grupos.

> Al agregar filtros a una consulta que incluye una cláusula group by , piense cuidadosamente sobre si el filtro actúa sobre datos brutos, en cuyo caso longs en la cláusula where , o en datos agrupados, en cuyo caso pertenece en la cláusula de tener .

Es posible, sin embargo, incluir las funciones de agregado en la que tiene cláusula, que no aparecen en la cláusula select , como lo demuestra lo siguiente:

```
mysql> SELECT product_cd, SUM (avail_balance) prod_balance
    -> DESDE cuenta
    -> DONDE estado = 'ACTIVO'
    -> GRUPO POR product_cd
    -> TENIENDO MIN (balance_disponible)> = 1000
```


re ldt W B kC

Página 14

```
    -> AND MAX (disponible_balance) <= 10000;
+ ------------ + -------------- +
| product_cd | prod_balance |
+ ------------ + -------------- +
| discos compactos | 19500.00 |
| MM | 17045.14 |
+ ------------ + -------------- +
2 filas en conjunto (0,00 seg)
```

Esta consulta genera saldos totales para cada producto activo, pero luego la condición de filtro en la cláusula de tener excluye todos los productos para los cuales el saldo mínimo es menor que $ 1,000 o el saldo máximo es mayor a $ 10,000.

Prueba tus conocimientos

Trabaje con los siguientes ejercicios para probar su comprensión de la agrupación y agregación de SQL características de puerta. Verifique su trabajo con las respuestas del Apéndice C.

Ejercicio 8-1

Construya una consulta que cuente el número de filas en la tabla de la cuenta .

Ejercicio 8-2

Modifique su consulta del Ejercicio 8-1 para contar el número de cuentas de cada cliente. Muestre el ID de cliente y el número de cuentas de cada cliente.

Ejercicio 8-3

Modifique su consulta del Ejercicio 8-2 para incluir solo aquellos clientes que tengan al menos dos cuentas.

Ejercicio 8-4 (Crédito adicional)

Encuentre el saldo total disponible por producto y rama donde hay más de uno cuenta por producto y sucursal. Ordene los resultados por saldo total (de mayor a menor).

re ldt W B kC