Procedimientos Almacenados en MySQL, condicionales

Sentencias de control condicionales.

Sentencia IF.

La declaración IF tiene tres formas:

- IF-THEN-ELSE simple,
- IF-THEN-ELSE,
- IF-THEN-ELSEIF-ELSE.

Declaración simple de MySQL IF-THEN

La instrucción IF-THEN permite ejecutar un conjunto de instrucciones SQL basadas en una condición específica. A continuación se ilustra la sintaxis de la instrucción IF-THEN :

```
IF   condition   THEN
  statements;
END IF ;
```

En esta sintaxis:

- Primero, especifique una condición para ejecutar el código entre IF-THEN y END IF. Si la condition evalúa como TRUE, se ejecutarán las declaraciones entre IF-THEN y END IF. De lo contrario, el control se pasa a la siguiente instrucción que sigue a END IF.
- En segundo lugar, especifique el código que se ejecutará si la condition evalúa como TRUE.

Utilizaremos la tabla de customers de la base de datos de muestra para la demostración:
Veamos el siguiente procedimiento almacenado GetCustomerLevel().

```
DELIMITER  $$

CREATE   PROCEDURE  GetCustomerLevel(
    IN   pCustomerNumber  INT ,
    OUT  pCustomerLevel   VARCHAR (20))
BEGIN
    DECLARE  credit  DECIMAL (10,2)  DEFAULT  0;

    SELECT  creditLimit
    INTO  credit
```

```
    FROM  customers
    WHERE  customerNumber = pCustomerNumber;

    IF  credit  >  50000  THEN
        SET  pCustomerLevel =  'PLATINUM' ;
    END IF ;
END $$

DELIMITER  ;
```

El procedimiento almacenado GetCustomerLevel() acepta dos parámetros: pCustomerNumber y pCustomerLevel .

- Primero, seleccione creditLimit del cliente especificado por pCustomerNumber de la tabla de customers y almacénelo en el credit variable local.
- Luego, establezca el valor para el parámetro OUT pCustomerLevel en PLATINUM si el límite de crédito del cliente es superior a 50,000 .

Esta declaración encuentra a todos los clientes que tienen un límite de crédito superior a 50,000 :

```
SELECT
   customerNumber,
   creditLimit
FROM
   customers
WHERE
   creditLimit  >  50000
ORDER BY
   creditLimit  DESC ;
```

Estas declaraciones llaman al procedimiento almacenado GetCustomerLevel() para el cliente 141 y muestran el valor del parámetro OUT pCustomerLevel:

```
CALL  GetCustomerLevel(141, @level);
SELECT  @level;
```

Debido a que el cliente 141 tiene un límite de crédito superior a 50,000 , su nivel se establece en PLATINUM como se esperaba.

Declaración MySQL IF-THEN-ELSE

En caso de que desee ejecutar otras declaraciones cuando la condition en la rama IF no se evalúa como TRUE, puede usar la declaración IF-THEN-ELSE siguiente manera:

```
IF   condition   THEN
  statements;
ELSE
   else - statements;
END IF ;
```

En esta sintaxis, si la condition evalúa como TRUE , se ejecutan las statements entre IF-THEN y ELSE. De lo contrario, se ejecutan las else-statements entre ELSE y END IF.
Modifiquemos el procedimiento almacenado GetCustomerLevel().
Primero, suelte el procedimiento almacenado GetCustomerLevel():

```
DROP   PROCEDURE  GetCustomerLevel;
```

Luego, cree el procedimiento almacenado GetCustomerLevel() con el nuevo código:

```
DELIMITER  $$

CREATE   PROCEDURE  GetCustomerLevel(
    IN   pCustomerNumber  INT ,
    OUT  pCustomerLevel   VARCHAR (20))
BEGIN
    DECLARE  credit  DECIMAL   DEFAULT  0;

    SELECT  creditLimit
    INTO  credit
    FROM  customers
    WHERE  customerNumber = pCustomerNumber;

    IF  credit  >  50000  THEN
        SET  pCustomerLevel =  'PLATINUM' ;
    ELSE
        SET  pCustomerLevel =  'NOT PLATINUM' ;
    END IF ;
END $$

DELIMITER  ;
```

En este nuevo procedimiento almacenado, incluimos la rama ELSE. Si el credit no es mayor a 50,000 , establecemos el nivel de cliente en NOT PLATINUM en el bloque entre ELSE y END IF.
Esta consulta busca clientes que tienen un límite de crédito menor o igual a 50,000:

```
SELECT
   customerNumber,
   creditLimit
FROM
   customers
WHERE
   creditLimit  < = 50000
ORDER BY
   creditLimit  DESC ;
```

Las siguientes declaraciones llaman al procedimiento almacenado para el número de cliente 447 y muestran el valor del parámetro OUT pCustomerLevel:

```
CALL  GetCustomerLevel(447, @level);
SELECT  @level;
```

El límite de crédito del cliente 447 es inferior a 50,000 , por lo tanto, la declaración en la rama ELSE se ejecuta y establece el valor del parámetro OUT pCustomerLevel en NOT PLATINUM .

Sentencia MySQL IF-THEN-ELSEIF-ELSE

Si desea ejecutar sentencias condicionalmente basadas en múltiples condiciones, use la siguiente IF-THEN-ELSEIF-ELSE :

```
 IF   condition   THEN
   statements;
 ELSEIF   elseif - condition   THEN
    elseif - statements;
...
 ELSE
    else - statements;
 END IF ;
```

En esta sintaxis, si la condition evalúa como TRUE , se ejecutan las statements en la rama IF-THEN ; de lo contrario, se evalúa la siguiente elseif-condition .
Si la elseif-condition evalúa como TRUE , se ejecuta la elseif-statement ; de lo contrario, se evalúa la siguiente elseif-condition .
La IF-THEN-ELSEIF-ELSE puede tener múltiples ramas ELSEIF .
Si ninguna condición en IF y ELSE IF evalúa como TRUE , se ejecutarán las else-statements en la rama ELSE .
Modificaremos el procedimiento almacenado GetCustomerLevel() para usar la instrucción IF-THEN-ELSEIF-ELSE .

- Primero, elimina el procedimiento almacenado GetCustomerLevel() :

```
DROP   PROCEDURE  GetCustomerLevel;
```

- Luego, cree el nuevo procedimiento almacenado GetCustomerLevel() que utiliza la instrucción IF-THEN-ELSEIF-ELSE.

```
DELIMITER  $$

CREATE   PROCEDURE  GetCustomerLevel(
    IN   pCustomerNumber  INT ,
    OUT  pCustomerLevel   VARCHAR (20))
BEGIN
    DECLARE  credit  DECIMAL   DEFAULT  0;

    SELECT  creditLimit
    INTO  credit
    FROM  customers
    WHERE  customerNumber = pCustomerNumber;

    IF  credit  >  50000  THEN
        SET  pCustomerLevel =  'PLATINUM' ;
    ELSEIF  credit  < = 50000  AND  credit  >  10000  THEN
        SET  pCustomerLevel =  'GOLD' ;
    ELSE
        SET  pCustomerLevel =  'SILVER' ;
```

```
    END IF ;
END  $$

DELIMITER  ;
```

En este procedimiento almacenado:

- Si el crédito es mayor a 50,000 , el nivel del cliente es PLATINUM .
- Si el crédito es menor o igual a 50,000 y mayor a 10,000 , entonces el nivel de cliente es GOLD .
- De lo contrario, el nivel del cliente es SILVER .

Estas declaraciones llaman al procedimiento almacenado GetCustomerLevel() y muestran el nivel del cliente 447 :

```
CALL  GetCustomerLevel(447, @level);
SELECT  @level;
```

Si prueba el procedimiento almacenado con el cliente que tiene un límite de crédito de 10000 o menos, obtendrá el resultado como SILVER.

Sentencia CASE

Además de la declaración IF , MySQL proporciona una declaración condicional alternativa llamada declaración CASE para construir declaraciones condicionales en procedimientos almacenados. Las declaraciones CASE hacen que el código sea más legible y eficiente.
La declaración CASE tiene dos formas: CASE simple y declaraciones CASE buscadas.
Tenga en cuenta que si desea agregar la lógica if-else a una declaración SQL, use la expresión CASE que es diferente de la declaración CASE descrita en este tutorial.

Sentencia CASE sencilla

La siguiente es la sintaxis básica de la declaración CASE simple:

```
CASE  case_value
   WHEN  when_value1  THEN  statements
   WHEN  when_value2  THEN  statements
  ...
  [ ELSE   else - statements]
END CASE ;
```

En esta sintaxis, la simple declaración CASE compara secuencialmente el valor case_value con when_value1 , when_value2 , ... hasta que encuentra que uno es igual. Cuando CASE encuentra un case_value igual a when_value , ejecuta statements en la correspondiente cláusula THEN.
Si CASE no puede encontrar ningún valor when_value igual al case_value, ejecuta las else-statements en la cláusula ELSE si la cláusula ELSE está disponible.
Cuando la cláusula ELSE no existe y el CASE no puede encontrar ningún valor when_value igual al case_value , emite un error:

```
Case not found for CASE statement
```

Tenga en cuenta que case_value puede ser un valor literal o una expresión. Las statements pueden ser una o más declaraciones SQL y no pueden tener una declaración cero.
Para evitar el error cuando case_value no es igual a when_value , puede usar un bloque BEGIN END vacío en la cláusula ELSE siguiente manera:

```
CASE  case_value
    WHEN  when_value1  THEN  ...
    WHEN  when_value2  THEN  ...
    ELSE
        BEGIN
        END ;
END CASE ;
```

La instrucción CASE (sencilla) prueba la igualdad (=), no puede usarla para probar la igualdad con NULL ; porque NULL = NULL devuelve FALSE .

Ejemplo de Sentencia CASE sencilla

El siguiente procedimiento almacenado ilustra cómo usar la declaración CASE simple:

```
 DELIMITER  $$

 CREATE   PROCEDURE  GetCustomerShipping(
     IN   pCustomerNUmber  INT ,
     OUT  pShipping        VARCHAR (50)
)
 BEGIN
     DECLARE  customerCountry  VARCHAR (100);

 SELECT
    country
 INTO  customerCountry  FROM
    customers
 WHERE
    customerNumber = pCustomerNUmber;

     CASE  customerCountry
         WHEN    'USA'   THEN
            SET  pShipping =  '2-day Shipping' ;
         WHEN   'Canada'   THEN
            SET  pShipping =  '3-day Shipping' ;
         ELSE
            SET  pShipping =  '5-day Shipping' ;
     END CASE ;
 END $$

 DELIMITER  ;
```

Cómo funciona.
En el procedimiento almacenado:

El procedimiento almacenado GetCustomerShipping() acepta dos parámetros: pCustomerNumber como parámetro IN y pShipping como parámetro OUT.

- Primero, seleccione el país del cliente de la tabla de customers por el número de cliente de entrada.
- En segundo lugar, use la declaración CASE simple para determinar el tiempo de envío según el país del cliente. Si el cliente se ubica en USA el tiempo de envío es de 2 días (2-day shipping). Si el cliente se ubica en Canada, el tiempo de envío es de 3 días (3-day shipping). Para los clientes de otros países son 5 días (5-day shipping).
- El siguiente diagrama de flujo muestra la lógica de la declaración CASE para determinar el tiempo de envío:

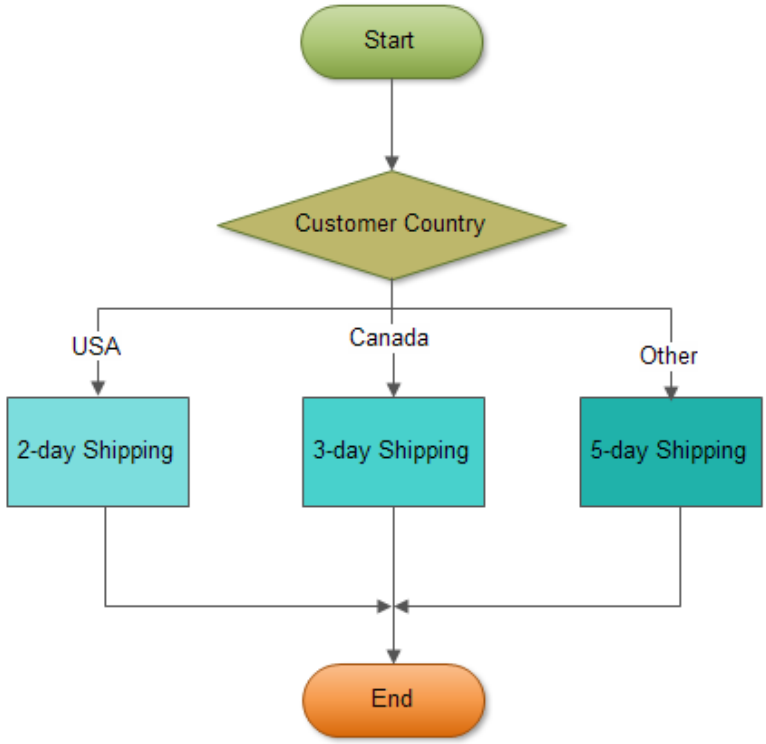

Figura 1: Diagrama de flujo para una sentencia CASE

Esta declaración llama al procedimiento almacenado y pasa el número de cliente 112:

```
CALL  GetCustomerShipping(112,@shipping);
```

La siguiente declaración devuelve el tiempo de envío del cliente 112:

```
SELECT  @shipping;
```

Aquí está la salida:

@shipping
2 - day Shipping

1 row in set (0.00 sec)

- :IGNORE: # C-u C-c - and C-c -

Sentencia CASE buscado (searched)

El CASE simple sólo permite comparar un valor con un conjunto de valores distintos.
Para realizar coincidencias más complejas, como rangos, utilice la instrucción CASE buscada. La declaración CASE buscada es equivalente a la declaración IF, sin embargo, es mucho más legible que la declaración IF.
Aquí está la sintaxis básica de la declaración CASE buscada:

```
CASE
    WHEN  search_condition1  THEN  statements
    WHEN  search_condition1  THEN  statements
   ...
   [ ELSE   else - statements]
END CASE ;
```

En esta sintaxis, el CASE buscado evalúa cada search_condition de search_condition en la cláusula WHEN hasta que encuentra una condición que se evalúa como TRUE, luego ejecuta las statements cláusula THEN correspondiente.
Si no search_condition evalúa como TRUE, el CASE ejecutará else-statements en la cláusula ELSE si hay una cláusula ELSE disponible.
Similar a la simple declaración CASE, si no especifica una cláusula ELSE y ninguna condición es TRUE, MySQL genera el mismo error:

```
Case not found for CASE statement
```

MySQL tampoco le permite tener sentencias vacías en la cláusula THEN o ELSE. Si no desea manejar la lógica en la cláusula ELSE tiempo que evita que MySQL search_condition un error en caso de que ninguna search_condition sea verdadera, puede usar un bloque BEGIN END vacío en la cláusula ELSE.

Ejemplo de sentencia CASE buscado

El siguiente ejemplo muestra cómo utilizar una declaración CASE buscada para encontrar el nivel de cliente SILVER, GOLD o PLATINUM función del límite de crédito del cliente.

```
 DELIMITER  $$

 CREATE   PROCEDURE  GetDeliveryStatus(
     IN  pOrderNumber INT,
     OUT  pDeliveryStatus  VARCHAR (100)
)
 BEGIN
     DECLARE waitingDay INT DEFAULT 0;
```

```
    SELECT
        DATEDIFF (requiredDate, shippedDate)
    INTO  waitingDay
    FROM  orders
    WHERE  orderNumber = pOrderNumber;

    CASE
        WHEN  waitingDay = 0  THEN
            SET  pDeliveryStatus =  'On Time' ;
        WHEN  waitingDay  > = 1  AND  waitingDay  <  5  THEN
            SET  pDeliveryStatus =  'Late' ;
        WHEN  waitingDay  > = 5  THEN
            SET  pDeliveryStatus =  'Very Late' ;
        ELSE
            SET  pDeliveryStatus =  'No Information' ;
    END CASE ;
END $$
DELIMITER  ;
```

Cómo funciona.
El procedimiento almacenado GetDeliveryStatus() acepta un número de pedido como un parámetro IN y devuelve el estado de entrega como un parámetro OUT .

- Primero, calcule el número de días entre la fecha requerida y la fecha de envío.
- En segundo lugar, determine el estado de entrega en función del número de días de espera utilizando la declaración CASE buscada:
    - Si el número de días de espera es cero, la entrega es puntual.
    - Cuando el número de días de espera es entre 1 y 5, la entrega se retrasa.
    - Cuando el número de días de espera es superior a 5 días, la entrega es muy tardía.
    - Si el número de días de espera es NULL o no, la entrega tiene el estado de ninguna información especificada en la cláusula ELSE .

Esta declaración utiliza el procedimiento almacenado GetDeliveryStatus() para obtener el estado de entrega del pedido 10100:

```
CALL  GetDeliveryStatus(10100,@delivery);
```

MySQL CASE vs. IF

Tanto las declaraciones IF como CASE permiten ejecutar un bloque de código basado en una condición específica. Elegir entre IF o CASE veces es solo una cuestión de preferencia personal. Aquí hay algunas pautas:

- Una declaración CASE simple es más legible y eficiente que una declaración IF cuando compara una sola expresión con un rango de valores únicos.

- Cuando verifica expresiones complejas basadas en múltiples valores, la instrucción IF es más fácil de entender.
- Si usa la declaración CASE , debe asegurarse de que al menos una de las condiciones CASE coincida. De lo contrario, debe definir un controlador de errores para detectar el error. Tenga en cuenta que no tiene que hacer esto con la instrucción IF .
- En algunas situaciones, puede usar IF y CASE para hacer que el código sea más legible y eficiente.