Material descargable

MySQL: Operadores de SubConsultas

Las subconsultas desempeñan un papel fundamental al permitirnos realizar consultas más complejas.

En este documento, analizaremos a fondo los operadores ANY, SOME y ALL, desentrañando sus diferencias y descubriendo cómo aprovechar al máximo su potencial.

A su vez, con el uso de tablas derivadas aprenderemos a estructurar consultas complejas de manera eficiente.

Estas técnicas avanzadas pueden elevar tus habilidades de consulta, permitiéndote abordar desafíos de análisis de datos con elegancia y precisión en entornos MySQL.

Subconsulta ANY y SOME

La palabra clave ANY, que debe seguir a un operador de comparación, significa "devolver TRUE si la comparación es TRUE para CUALQUIERA de los valores que devuelve la subconsulta". Por ejemplo:

```
SELECT c1 FROM t1 WHERE c1 > ANY (SELECT c1 FROM t2);
```

Supongamos que hay una fila en la tabla t1 que contiene (10). La expresión es TRUE si la tabla t2 contiene (21,14,7) porque hay un valor 7 en t2 que es menor que 10. La expresión es FALSE si la tabla t2 contiene (20,10), o si la tabla t2 está vacía. La expresión es NULL si la tabla t2 contiene (NULL,NULL,NULL).

La palabra SOME es un alias de ANY. Por lo tanto, estas dos declaraciones son equivalentes:

```
SELECT c1 FROM t1 WHERE c1 <> ANY (SELECT c1 FROM t2);
SELECT c1 FROM t1 WHERE c1 <> SOME (SELECT c1 FROM t2);
```

El uso de la palabra "SOME" es poco común, su uso busca hacer más comprensivas algunas consultas. En la mayoría de los casos, la frase en inglés "a is not equal to any b" (a no es igual a ningún b) se interpreta como "there is no b which is equal to a" (no existe ningún b que sea igual a a). Sin embargo, esta no es la interpretación que sigue la sintaxis SQL. En SQL, la sintaxis significa "there is some b to which a is not equal" (existe al menos un b al que a no es igual") El uso de "<> SOME" en lugar de "<> ANY" ayuda a garantizar que todos comprendan el verdadero significado de la consulta. En otras palabras, "<> SOME" enfatiza que se busca al menos una coincidencia en lugar de ninguna, lo que puede evitar confusiones al interpretar la consulta, como se ilustra en la siguiente consulta:

```
SELECT * FROM tabla_a
WHERE valor_a <> SOME (SELECT valor_b FROM tabla_b);
```

Esta consulta busca filas en tabla_a donde el valor de valor_a no sea igual a al menos un valor en la columna valor_b de `tabla_b.

Subconsulta ALL

La palabra "ALL", que debe seguir a un operador de comparación, significa "devolver TRUE si la comparación es TRUE para TODOS los valores la subconsulta devuelve".

```
SELECT c1 FROM t1 WHERE c1 > ALL (SELECT c1 FROM t2);
```

Supongamos que hay una fila en la tabla t1 que contiene (10). La expresión es VERDADERA si la tabla t2 contiene (-5, 0, +5) porque 10 es mayor que los tres valores en t2. La expresión es FALSA si la tabla t2 contiene (12, 6, NULL, -100) porque hay un solo valor, 12 en la tabla t2, que es mayor que 10. La expresión es desconocida (es decir, NULL) si la tabla t2 contiene (0, NULL, 1).

La siguiente expresión es VERDADERA cuando la tabla t2 está vacía:

```
SELECT * FROM t1 WHERE 1 > ALL (SELECT c1 FROM t2);
```

Pero esta expresión es NULL cuando la tabla t2 está vacía:

```
SELECT * FROM t1 WHERE 1 > (SELECT c1 FROM t2);
```

Además, la siguiente expresión es NULL cuando la tabla t2 está vacía por usar una función:

```
SELECT * FROM t1 WHERE 1 > ALL (SELECT MAX(c1) FROM t2);
```

En general, las tablas que contienen valores NULL y las tablas vacías son casos "extremos". Al escribir subconsultas, siempre considera si has tenido en cuenta esas dos posibilidades.

Subconsultas de fila

Las subconsultas escalares o de columna devuelven un único valor. Una subconsulta de fila es una variante de subconsulta que devuelve una sola fila y, por lo tanto, puede devolver más de una columna:

```
SELECT * FROM t1
  WHERE (col1, col2) = (SELECT col3, col4 FROM t2 WHERE id = 10);
SELECT * FROM t1
  WHERE ROW(col1, col2) = (SELECT col3, col4 FROM t2 WHERE id = 10);
```

Para ambas consultas, si la tabla t2 contiene una sola fila con id = 10, la subconsulta devuelve una sola fila. Si esta fila tiene valores col3 y col4 iguales a los valores col1 y col2 de cualquier fila en t1, la expresión WHERE es VERDADERA y cada consulta devuelve esas filas de t1. Si los valores col3 y col4 de la fila t2 no son iguales a los valores col1 y col2 de ninguna fila de t1, la expresión es FALSA y la consulta devuelve un conjunto de resultados vacío. La expresión es desconocida (es decir, NULL) si la subconsulta no produce filas. Se produce un error si la subconsulta produce múltiples filas porque una subconsulta de fila puede devolver como máximo una fila.

💡*Las expresiones (1,2) y ROW(1,2) a veces se llaman constructores de filas y son equivalentes.*

Los constructores de filas son legales en otros contextos. Por ejemplo, las siguientes dos declaraciones son equivalentes desde el punto de vista semántico (y son tratadas de la misma manera por el optimizador):

```
SELECT * FROM t1 WHERE (col1, col2) = (1, 1);
SELECT * FROM t1 WHERE col1 = 1 AND col2 = 1;
```

La siguiente consulta responde a la solicitud "encontrar todas las filas en la tabla t1 que también existen en la tabla t2":

```
SELECT col1, col2, col3
  FROM t1
  WHERE (col1, col2, col3) IN
        (SELECT col1, col2, col3 FROM t2);
```

Subconsultas correlacionadas

Una subconsulta correlacionada es una subconsulta que contiene una referencia a una tabla que aparece en la consulta externa. Por ejemplo:

```
SELECT * FROM t1
  WHERE col1 = ANY (SELECT col1 FROM t2
                    WHERE t2.col2 = t1.col2);
```

Observa que la subconsulta contiene una referencia a una columna de t1, aunque la cláusula FROM de la subconsulta no menciona una tabla t1. Por lo tanto, MySQL busca fuera de la subconsulta y encuentra t1 en la consulta externa.

Supongamos que la tabla t1 contiene una fila donde col1 = 5 y col2 = 6; mientras tanto, la tabla t2 contiene una fila donde col1 = 5 y col2 = 7. La expresión simple "...WHERE col1 = ANY (SELECT column1 FROM t2)" sería VERDADERA, pero en este ejemplo, la cláusula WHERE dentro de la subconsulta es FALSA (porque (5,6) no es igual a (5,7)), por lo que la expresión en su conjunto es FALSA.

Regla de ámbito: MySQL evalúa de adentro hacia afuera. Por ejemplo:

```
SELECT col1 FROM t1 AS x
  WHERE x.col1  = (SELECT col1 FROM t2 AS x
    WHERE x.col1  = (SELECT col1 FROM t3
      WHERE x.col2  = t3.col1 ));
```

En esta declaración, x.col2 debe ser una columna en la tabla t2 porque "SELECT col1 FROM t2 AS x ..". renombra t2. No es una columna en la tabla t1 porque "SELECT col1 FROM t1 ..." es una consulta externa que está más alejada.

Tabla derivada

Una tabla derivada es una expresión que genera una tabla dentro del alcance de la cláusula FROM de una consulta. Por ejemplo, una subconsulta en la cláusula FROM de una declaración SELECT es una tabla derivada:

```
SELECT ... FROM (subconsulta) [AS] nombre_tabla ...
```

La cláusula [AS] nombre_tabla es obligatoria porque cada tabla en una cláusula FROM debe tener un nombre. Cualquier columna en la tabla derivada debe tener nombres únicos. Alternativamente, nombre_tabla puede ir seguido de una lista entre paréntesis de nombres para las columnas de la tabla derivada:

```
SELECT ... FROM (subconsulta) [AS] nombre_tabla (lista_col) ...
```

El número de nombres de columna debe ser igual al número de columnas de la tabla.

Supongamos que tienes esta tabla:

```
CREATE TABLE t1 (c1 INT, c2 CHAR(5), c3 FLOAT);
```

Así es como se utiliza una subconsulta en la cláusula FROM, usando la tabla de ejemplo:

```
INSERT INTO t1 VALUES (1,'1',1.0);
INSERT INTO t1 VALUES (2,'2',2.0);
SELECT cb1, cb2, cb3
  FROM (SELECT c1 AS cb1, c2 AS cb2, c3*2 AS cb3 FROM t1) AS cb
  WHERE cb1 > 1;

#Resultado:
+------+------+------+
| cb1  | cb2  | cb3  |
+------+------+------+
|    2 | 2    |    4 |
+------+------+------+
```

Aquí tienes otro ejemplo: Supongamos que deseas conocer el promedio de un conjunto de sumas para una tabla agrupada. Esto no funcionaría:

```
SELECT AVG(SUM(column1)) FROM t1 GROUP BY column1;
```

Sin embargo, esta consulta sí proporciona la información deseada:

```
SELECT AVG(sum_col1)
  FROM (SELECT SUM(col1) AS sum_col1
        FROM t1 GROUP BY col1) AS t1;
```

Observa que el nombre de columna utilizado dentro de la subconsulta (sum_col1) es reconocido en la consulta externa.

Subconsulta IN vs ANY y ALL

Cuando se utiliza con una subconsulta, la cláusula IN es equivalente a "= ANY".

```
SELECT c1 FROM t1 WHERE c1 = ANY (SELECT c1 FROM t2);
SELECT c1 FROM t1 WHERE c1 IN (SELECT c1 FROM t2);
```

IN y = ANY no son sinónimos cuando se utilizan con una lista de expresiones. IN puede tomar una lista de expresiones, pero = ANY no puede.

```
SELECT c1 FROM t1 WHERE c1 IN (5, 10, 20);
# Pero si lo intentamos con ANY no funcionará:
SELECT c1 FROM t1 WHERE c1 = ANY (5, 10, 20);
```

NOT IN no es equivalente a <> ANY, es equivalente a <> ALL.

```
SELECT c1 FROM t1 WHERE c1 <> ALL (SELECT c1 FROM t2);
SELECT c1 FROM t1 WHERE c1 NOT IN (SELECT c1 FROM t2);
```