@author: aar.velasquez@gmail.com @date: 2020/10/14

Capítulo 5: Consultar varias tablas

¿Qué es una unión?

```
Es la unión de una o mas tablas, esta unión se hace mediante un 'puente'que
relaciona dichas tablas.
```

Producto cartesiano

Aquí hay una consulta que recupera primer nombre y apellidos junto con el nombre del departamento, con una cláusula **FROM** nombrando ambas tablas separados por la palabra clave de **JOIN** :

```
mysql> SELECT e.fname, e.lname, d.name
    -> FROM employee e JOIN department d;

----
Debido a que la consulta no especificó cómo se deben unir las dos tablas,
la base de datos
servidor generó el producto cartesiano, que es cada permutación de las dos
tablas
(18 empleados × 3 departamentos = 54 permutaciones). Este tipo de
combinación se conoce como
unión cruzada, y rara vez se usa (al menos a propósito).
----
```

Uniones internas (Inner Joins)

Para modificar la consulta anterior de modo que solo se incluyan 18 filas en el conjunto de resultados (una para cada empleado), debe describir cómo se relacionan las dos tablas. por lo que esta información debe agregarse a la subcláusula **ON** de la cláusula **FROM** :

```
mysql> SELECT e.fname, e.lname, d.name
    -> FROM employee e JOIN department d
        ->ON e.dept_id = d.dept_id;
```

En el ejemplo anterior, no especifiqué en la cláusula from qué tipo de combinación usar. Sin embargo, cuando desee unir dos tablas utilizando una combinación interna, debe explícitamente especifique esto en su cláusula from ; aquí está el mismo ejemplo, con la adición de la combinación tipo (tenga en cuenta la palabra clave INNER):

```
mysql> SELECT e.fname, e.lname, d.name
    -> FROM employee e INNER JOIN department d
    -> ON e.dept_id = d.dept_id;
```

Si no especifica el tipo de combinación, el servidor realizará una combinación interna de forma predeterminada. Sin embargo, como verá más adelante en el libro, hay varios tipos de combinaciones, por lo que debe Adquiera el hábito de especificar el tipo exacto de unión que necesita.

Si los nombres de las columnas utilizadas para unir las dos tablas son idénticos, lo que es cierto en la consulta anterior, puede utilizar la subcláusula using en lugar de la subcláusula on , como en:

```
mysql> SELECT e.fname, e.lname, d.name
    -> FROM employee e INNER JOIN department d
        ->USING (dept_id);
```

La sintaxis de unión ANSI

Todas las principales bases de datos (Oracle Database, Microsoft SQL Server, MySQL, IBM DB2 Universal Database y Sybase Adaptive Server) han adoptado la sintaxis de combinación SQL92. Debido a que la mayoría de estos servidores desde antes del lanzamiento de la especificación SQL92.

la sintaxis de unión ANSI tiene las siguientes ventajas:

```
• Las condiciones de combinación y las condiciones de filtro se separan en
dos cláusulas diferentes (la
subcláusula ON y la cláusula where , respectivamente), lo que facilita
entender la consulta.

• Las condiciones de combinación para cada par de tablas están contenidas
en su propia cláusula ON ,
lo que hace menos probable que parte de una combinación se omita por error.

• Las consultas que utilizan la sintaxis de combinación SQL92 son portables
entre servidores de bases de datos,
mientras que la sintaxis anterior es ligeramente diferente en los
diferentes servidores.
```

Unir tres o más tablas

Aquí está otro ejemplo de una consulta con una combinación de dos tablas:

```
mysql> SELECT a.account_id, c.fed_id
    -> FROM account a INNER JOIN customer c
    ->      ON a.cust_id = c.cust_id
    -> WHERE c.cust_type_cd = 'B';

+------------+------------+
| account_id | fed_id
|
+------------+------------+
|
    24       | 04-1111111 |
|
    25       | 04-1111111 |
|
    27       | 04-2222222 |
|
    28       | 04-3333333 |
|
    29       | 04-4444444 |
+------------+------------+
Sin embargo, si agrega el empleado tabla a la consulta para recuperar
también el nombre del cajero que abrió cada cuenta, tiene el siguiente
aspecto:

mysql> SELECT a.account_id, c.fed_id, e.fname, e.lname
-> FROM account a INNER JOIN customer c
    ->ON a.cust_id = c.cust_id
    ->INNER JOIN employee e
    ->ON a.open_emp_id = e.emp_id
-> WHERE c.cust_type_cd = 'B';

+------------+------------+---------+---------+
| account_id  | fed_id  | fname   | lname   |
+------------+------------+---------+---------+
|
        24 | 04-1111111 | Theresa | Markham |
|
        25 | 04-1111111 | Theresa | Markham |
|
        27 | 04-2222222 | Paula | Roberts   |
|
        28 | 04-3333333 | Theresa | Markham |
|
         29 | 04-4444444 | John   | Blake   |
+------------+------------+---------+---------+
```

Usar subconsultas como tablas

busque todas las cuentas abiertas por cajeros experimentados actualmente asignados al Woburn branch) que une la tabla de cuentas a las subconsultas de la rama y mesas de empleados :

```
1 SELECT a.account_id, a.cust_id, a.open_date, a    product_cd
2 FROM account a INNER JOIN
3 (SELECT emp_id, assigned_branch_id
4 FROM employee
5 WHERE start_date < '2007-01-01'
6 AND (title = 'Teller' OR title = 'Head Teller')) e
7 ON a.open_emp_id = e.emp_id
8 INNER JOIN
9 (SELECT branch_id
10 FROM branch
11 WHERE name = 'Woburn Branch') b
12ON e.assigned_branch_id = b.branch_id;
```

- La primera subconsulta, que comienza en la línea 3 y recibe el alias e , encuentra todos los cajeros.
- La segunda subconsulta, que comienza en la línea 9 y recibe el alias b , encuentra el Identificación de la sucursal de Woburn. Primero, la mesa de cuentas está unida al cajero experimentado. subconsulta usando el ID de empleado y luego la tabla resultante se une al Woburn subconsulta de rama usando el ID de rama. Los resultados son los mismos que los del anterior versión de la consulta (pruébelo y compruébelo usted mismo), pero las consultas se ven muy diferentes de unos y otros.

Usar la misma tabla dos veces

Para que esto funcione, tendrá que dar a cada instancia de la tabla de rama un alias diferente para que el servidor sabe a cuál te refieres en las distintas cláusulas, como en:

```
mysql> SELECT a.account_id, e.emp_id,
    ->b_a.name open_branch, b_e.name emp_branch
    -> FROM account a INNER JOIN branch b_a
    ->ON a.open_branch_id = b_a.branch_id
    ->INNER JOIN employee e
    ->ON a.open_emp_id = e.emp_id
    ->INNER JOIN branch b_e
    -> ON e.assigned_branch_id = b_e.branch_id
-> WHERE a.product_cd = 'CHK';
```

Autouniones

Puede realmente unir una mesa a sí mismo. Esto puede parecer extraño al principio, pero hay razones válidas para hacerlo.

```
mysql> SELECT e.fname, e.lname,
    ->e_mgr.fname mgr_fname, e_mgr.lname mgr_lname
```

```
    -> FROM employee e INNER JOIN employee e_mgr
    ->ON e.superior_emp_id = e_mgr.emp_id;
```

Equi-Joins versus no Equi-Joins

Un equi-join siempre que emplea un signo igual, como en:

```
ON e.assigned_branch_id = b.branch_id
```

Si bien la mayoría de sus consultas emplearán equi-joins, también puede unir sus tablas a través de rangos de valores, que se conocen como no equi-uniones. He aquí un ejemplo de consulta que se une por un rango de valores:

```
SELECT e.emp_id, e.fname, e.lname, e.start_date
    FROM employee e INNER JOIN product p
    ON e.start_date >= p.date_offered
    AND e.start_date <= p.date_retired
WHERE p.name = 'no-fee checking';
```

Condiciones de unión versus condiciones de filtro

Ahora está familiarizado con el concepto de que las condiciones de unión pertenecen a la subcláusula on , mientras que las condiciones de filtro pertenecen a la cláusula where .