UD3

CONSULTA Y MODIFICACIÓN DE BASES DE DATOS

MP_0484

Bases de Datos

3.6 Consultas asociadas

Introducción

Como hemos visto anteriormente, podemos usar cláusulas JOIN para obtener datos de varias tablas con una sola consulta SELECT. Sin embargo, si bien las uniones INNER/LEFT/RIGHT/FULL pueden ser útiles, no siempre son la respuesta a las consultas más complejas, aquellas que requieren recopilar información concreta de ciertas tablas. En este capítulo aprenderemos a asociar tablas para sacar el máximo partido a nuestras consultas

Es fundamental, tal y como venimos repitiendo desde el inicio del curso, entender a la perfección el diagrama relacional de la base de datos y especialmente las relaciones entre tablas, por lo que antes de comenzar se recomienda echar un vistazo al diagrama de ingeniería inversa

Consultas asociadas

En primer lugar, debemos identificar qué tablas están involucradas en nuestra consulta. Por ejemplo, si necesitamos averiguar el nombre del contable mejor pagado, sabemos que tendremos que trabajar con tablas de títulos, empleados y salarios (figura 1).

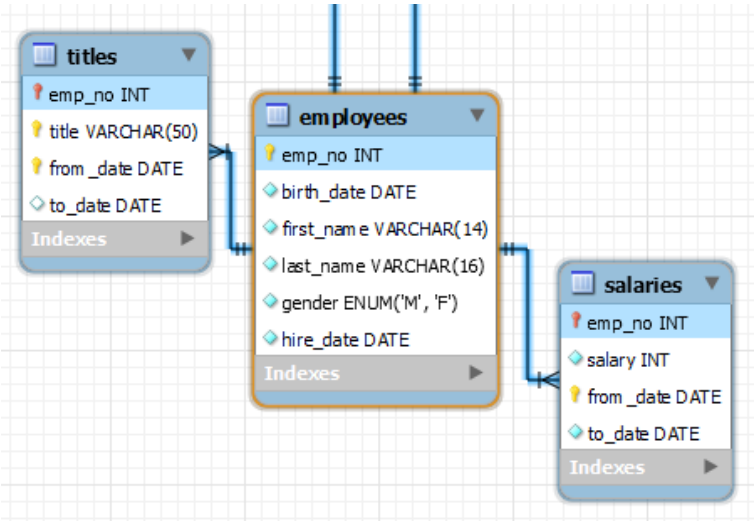

Figura 1. Base de datos de ejemplo para la consulta

- **Títulos**, ya que debemos buscar a aquellos empleados cuyo título es "contable".
- **Salarios**, ya que buscamos el salario más alto.
- **Empleados**, ya que es la tabla que se encuentra entre los salarios y los títulos.

A continuación, tenemos que identificar qué columnas conectan esas tablas.

En nuestro ejemplo, todas las tablas están relacionadas por la columna emp_no, que es el PK para los empleados y FK para los títulos y salarios. Así que ahora, tenemos un enlace entre las 3 tablas.

Antes de continuar, debemos entender que no es necesario que todas las tablas involucradas estén vinculadas por la misma columna (en nuestro ejemplo, emp_no). Siempre que tengamos una cadena de columnas (generalmente PK y FK) que vinculen todas las tablas, estamos listos para realizar nuestra consulta.

Vamos a implementar dos posibles soluciones a esta consulta (figuras 2 y 3).

```
SELECT e.emp_no, e.first_name, e.last_name, e.gender, t.title, s.salary
FROM employees e
JOIN titles t ON e.emp_no = t.emp_no
JOIN salaries s ON e.emp_no = s.emp_no
WHERE t.title = 'Accountant'
AND s.salary = (
    SELECT MAX(salary)
    FROM salaries
    JOIN titles ON salaries.emp_no = titles.emp_no
    WHERE titles.title = 'Accountant'
);
```

Figura 2. Script a)

```
SELECT e.emp_no, e.first_name, e.last_name, e.gender, t.title, s.salary
FROM employees e
JOIN titles t ON e.emp_no = t.emp_no
JOIN salaries s ON e.emp_no = s.emp_no
WHERE t.title = 'Accountant'
ORDER BY s.salary DESC
LIMIT 1;
```

Figura 3. Script b)

Aunque este no sea el objetivo de esta unidad, hay que tener en cuenta que los diferentes enfoques, aunque todos puedan ser correctos, pueden marcar la diferencia en lo que respecta al rendimiento. Veamos en la figura 4 los tiempos de ejecución de ambos códigos.

	#	Time	Action	Message	Duration / Fetch
✔	31	13:13:09	SELECT e.emp_no, e.first_name, e.last_name, e.gender, t.title, s.salary FROM emplo...	1 row(s) returned	0.797 sec / 0.000 sec
✔	32	13:13:36	SELECT e.emp_no, e.first_name, e.last_name, e.gender, t.title, s.salary FROM emplo...	1 row(s) returned	0.484 sec / 0.000 sec

Figura 4. Tiempos de ejecución de las consultas

Como se puede ver, la primera consulta tarda casi el doble que la segunda, mientras que ambas producen un resultado correcto. Esto puede producir un grave impacto cuando se consideran sistemas grandes.

Añadiendo complejidad

Pensemos ahora que cada departamento tiene un coche (que es conducido por el gerente del departamento). ¿Cómo podemos saber quién conduce el coche con matrícula 9891CGL? Necesitamos asociar teniendo en cuenta 2 enlaces diferentes (figura 5).

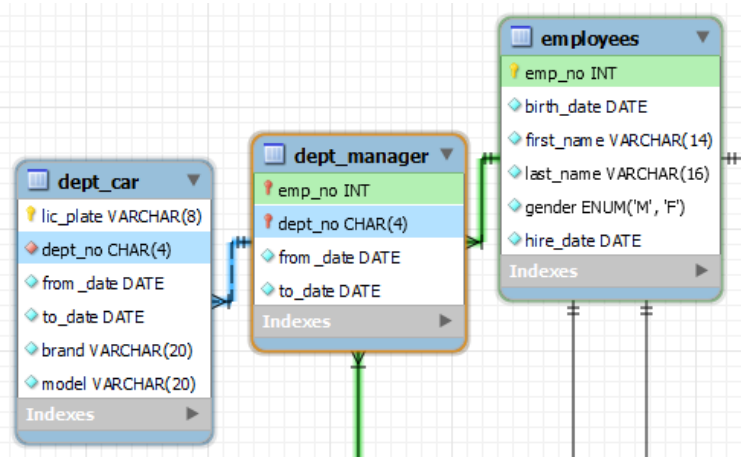

Figura 5. Base de datos de ejemplo para la consulta

- **Dept_car** y **dept_manager** están vinculados por dept_no, que es FK para ambos.
- **Los empleados** y **dept_manager** están vinculados por emp_no, que es PK para los empleados y FK para dept_manager.

Por tanto, vemos a continuación una posible implementación de la consulta (figura 6).

```
SELECT e.emp_no, e.first_name, e.last_name, dc.lic_plate, dc.brand, dc.model
FROM dept_manager dm
JOIN dept_car dc ON dm.dept_no = dc.dept_no
JOIN employees e ON dm.emp_no = e.emp_no
WHERE dc.lic_plate = '9891CGL' AND dm.to_date = '9999-01-01';
```

Figura 6. Script de la consulta

Como podemos ver, hemos vinculado las tres tablas utilizando sus PK y FK y agregando filtros (solo para encontrar la matrícula coincidente y el gerente de departamento actual), lo que produce el siguiente resultado (figura 7):

	emp_no	first_name	last_name	lic_plate	brand	model
▶	110114	Isamu	Legleitner	9891CGL	BMW	520i

Figura 7. Resultado de la consulta

Buenas prácticas en las consultas

Ten en cuenta que en los ejemplos con los que estamos trabajando utilizamos unos pocos registros, pero lo normal es trabajar en una base de datos con miles de ellos. La degradación del rendimiento se produce a menudo si el mantenimiento de la base de datos no es el adecuado o si las consultas se pueden reescribir de forma más eficaz. Veamos a continuación algunas recomendaciones para garantizar la optimización de consultas:

- Evita el uso de funciones en WHERE, ORDER BY y GROUP BY.
- Evita el uso del carácter comodín *.
- Evita columnas innecesarias en la cláusula WHERE. No usemos SELECT *, las columnas innecesarias imponen cargas adicionales en la BD, lo que ralentiza no solo el SQL específico, sino todo el sistema.
- Utiliza INNER JOIN en lugar de OUTER JOIN, si es posible. La unión externa solo debe utilizarse si es necesario. Su uso limita las opciones de optimización de la base de datos, lo que suele dar como resultado una ejecución más lenta del SQL.
- UNION sólo debe utilizarse si es necesario. Utiliza UNION ALL en lugar de UNION, si es posible, ya que es mucho más eficaz, porque no tiene que comprobar las repetidas.
- Ten en cuenta el impacto del rendimiento que supone añadir la cláusula ORDER BY, ya que la base de datos necesita clasificar el conjunto de resultados, lo que se convierte en una de las operaciones más costosas de la ejecución de SQL.