

Planes de ejecución: Sentencia Explain

Índice

01 [Sentencia Explain](#)



01

Sentencia Explain

Introducción

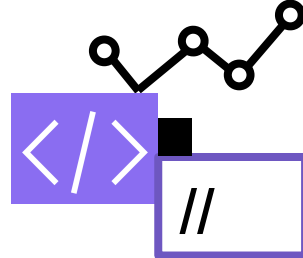
Cada vez que se ejecuta una consulta, sigue un Plan de Ejecución elaborado por el optimizador de consultas.

Es él quien, por ejemplo, define el orden de las tablas a consultar, y si va a ejecutar un índice creado o escanear la tabla.

La mayoría de las veces, el optimizador de consultas dibuja el mejor plan, que consume menos recursos o tiene más rendimiento.



Para analizar el plan de ejecución elegido por el optimizador, usamos la sentencia Explain.

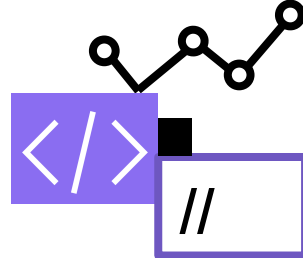


Analizando la consulta con la sentencia Explain

Analizaremos la siguiente consulta, utilizando la base de datos adventureworks:

SQL

```
SELECT contact.ContactId, contact.FirstName, contact.LastName,  
employee.Title  
FROM contact  
INNER JOIN employee on contact.ContactID = employee.ContactID  
where FirstName like 'C%';
```

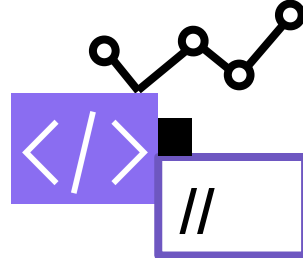


Analizando la consulta con la sentencia Explain

Esta consulta devolvió 12 registros. Veamos cómo el optimizador procesó esta consulta, usando la sentencia Explain

SQL

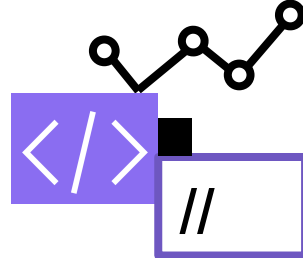
```
EXPLAIN SELECT contact.ContactId, contact.FirstName,  
contact.LastName, employee.Title  
FROM contact  
INNER JOIN employee on contact.ContactID = employee.ContactID  
where FirstName like 'C%';
```



Analizando la consulta con la sentencia Explain

Observemos que el optimizador trazó un plan diferente al de nuestra consulta. No siguió el orden que estipulamos en **FROM**, sino que ejecutó primero la tabla de empleados y luego el contacto.

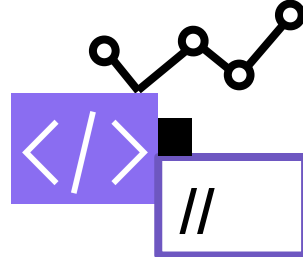
	id	select_type	table	partitions	type	possible_keys	key
▶	1	SIMPLE	employee	NULL	ALL	NULL	NULL
	1	SIMPLE	contact	NULL	eq_ref	PRIMARY	PRIMARY



Analizando la consulta con la sentencia Explain



En esta otra parte del **Explain**, podemos ver que el optimizador filtró una taza de 11.11% de la tabla contact, utilizando la cláusula **WHERE**.

key_len	ref	rows	filtered	Extra
NULL	NULL	290	100.00	NULL
4	adventureworks.employee.ContactID	1	11.11	Using where



Analizando la consulta con la sentencia Explain

Sin embargo, si la intención era forzar que esta consulta se ejecutará en el orden que definimos en **FROM**, podríamos usar **STRAIGHT_JOIN** en lugar de Inner **Join**. Esto nos haría ignorar el plan de ejecución del optimizador. Veamos la explicación de este cambio:

Result Grid Filter Rows: <input type="text"/> Export:  Wrap Cell Content: 												
	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	contact	NULL	ALL	PRIMARY	NULL	NULL	NULL	19855	11.11	Using where
	1	SIMPLE	employee	NULL	ALL	NULL	NULL	NULL	NULL	290	10.00	Using where; Using join buffer (hash join)

Ahora la consulta se ejecutó en el orden que estipulamos, sin pasar por el optimizador de MySQL.

¡Muchas gracias!