**Situación 1:**

Costo por:

* Id\_Cliente: 0,0032831
* Nombre\_Cliente: 0,0032842

1) La diferencia en el plan de ejecución es que la búsqueda por Id\_Cliente (clave primaria) es más rápida ya que las claves están indexadas. Además, si agregamos varios registros con el mismo apellido, la búsqueda por nombre del cliente arrojará varios resultados y tardará más en filtrarlos. Ej. 2 registros: 0,0032864.

Indexamos Nombre\_Cliente.

* Hay 4 clientes Perez, el costo es: 0,0032875.
* Hay 1 Lopez, el costo es el mismo: 0,0032875.

2) Después de la creación del índice los costos aumentaron.

Indexamos Nombre\_Cliente CLUSTERED.

* Hay 4 clientes Perez, el costo es: 0,0032864.
* Hay 1 Lopez, el costo es el mismo: 0,0032831.
* Por Id: 0,0065704

3) La diferencia es que los costos han bajado, ya que el CLUSTERED ordena físicamente los registros por Nombre\_Cliente.

4) Tener CLUSTERED por Nombre\_Cliente no empeora la búsqueda por Id\_Cliente ya que siempre es 1 sólo, con el índice va directo al registro. En nuestro caso sí empeoró porque tenemos pocos registros cargados en la tabla.

**Situación 2:**

1) El índice en factura sigue siendo Nro\_Factura, por lo tanto al buscar por cliente se va a recorrer toda la tabla comparando con el Id\_Cliente correspondiente. (fuerza bruta)

2) La solución sería crear un índice Id\_Cliente en la tabla facturas.

**Situación 3:**

1) Tabla Clientes: 9500 registros. 5 Categorías distintas 🡪 9500/5 = 1900 registros.

2) Tabla Productos: 25000 registros. 20 familias de productos 🡪 25000/20 = 1250 registros.

3) 12000\*(10-1)/(100-1) = 10909 registros. Esto es cantidad <= 10 🡪 120000 – 10909 = 109090 registros.

El cálculo termina ahí porque las consultas por clientes y producto tienen índices y van directo (costo 1, multiplicar por uno no suma nada)

**Situación 4:**

1) Esta petición (intersección más lo demás) va a devolver las personas que tiene una cuenta y a la vez un préstamo en “casa central”. Que Nombre sucursal sea clave primaria optimiza la operación where gracias a su índice.

2) Por cada persona en préstamo se busca si tiene una cuenta. La subconsulta en cuenta no es óptima porque al tener un “in” no se puede utilizar el indice. Se consulta varias veces para un cliente (en subconsulta)

3) Se realiza el producto cartesiano entre préstamo y cuenta, y luego selecciona por cliente y nombre de sucursal. El producto cartesiano genera una tabla muy grande.

Conclusión: 1 es más óptimo porque primero filtra y después une, caso contrario a 3

Ranking: 1 – 3 – 2