Optimización de consulta SQL

Consulta original

```sql
WITH ProductsOrder43676 AS (
    SELECT productid
    FROM AdventureWorks.sales.SalesOrderDetail
    WHERE salesorderid = 43676
),
CandidateOrders AS (
    SELECT salesorderid, productid
    FROM AdventureWorks.sales.SalesOrderDetail
    WHERE salesorderid <> 43676
)
SELECT salesorderid
FROM CandidateOrders
GROUP BY salesorderid
HAVING COUNT(DISTINCT productid) >= (
            SELECT COUNT(DISTINCT productid)
            FROM ProductsOrder43676
        )
   AND NOT EXISTS (
        SELECT productid
        FROM ProductsOrder43676
        EXCEPT
        SELECT productid
        FROM CandidateOrders c
        WHERE c.salesorderid = CandidateOrders.salesorderid
   );
```

Pasos de optimización

1. **Eliminación de CTE redundante:** Ya no es necesario el CTE `CandidateOrders`, podemos trabajar directamente sobre `SalesOrderDetail`.

2. **Materialización de productos originales:** Se separa en un CTE `OriginalProducts` para listar los `productid` únicos de la orden 43676.

3. **Cálculo único del conteo original:** Se introduce un CTE `OriginalCount` que almacena una sola vez el número total de productos distintos de la orden original.

4. **JOIN en lugar de EXCEPT/NOT EXISTS:** Al hacer un `JOIN` previo con `OriginalProducts`, reducimos filas a comprobar en el agrupamiento.

5. **Uso de `COUNT(DISTINCT)` en el `HAVING`:** Garantiza que cada pedido candidato cubra todos los productos sin necesidad de subconsultas anidadas adicionales.

6. **Índice sugerido:** Un índice sobre `(salesorderid, productid)` acelera tanto el `JOIN` como el `GROUP BY`.

Consulta optimizada

```
WITH OriginalProducts AS (
    SELECT DISTINCT productid
    FROM AdventureWorks.sales.SalesOrderDetail
    WHERE salesorderid = 43676
),
OriginalCount AS (
    SELECT COUNT(*) AS cnt
    FROM OriginalProducts
)
SELECT sod.salesorderid
FROM AdventureWorks.sales.SalesOrderDetail AS sod
  INNER JOIN OriginalProducts AS op
    ON sod.productid = op.productid
WHERE sod.salesorderid <> 43676
GROUP BY sod.salesorderid
HAVING COUNT(DISTINCT sod.productid) = (SELECT cnt FROM
   OriginalCount);
```

Sugerencias adicionales

- **Índice no clusterizado:**

```
CREATE NONCLUSTERED INDEX IX_SOD_Order_Product
  ON AdventureWorks.sales.SalesOrderDetail(salesorderid,
       productid);
```

- **Actualizar estadísticas:** Tras grandes cargas o limpiezas de datos, ejecutar `UPDATE STATISTICS` en la tabla para mantener buenos planes de ejecución.
- **Considerar partición (si la tabla es muy grande):** Particionar por rango de `salesorderid` ayuda a paralelizar búsquedas en entornos con alta concurrencia.