

INSTITUTO POLITECNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERIA

Y TECNOLGIAS AVANZADAS - IPN

**MATERIA**

*Bases de Datos Distribuidas*

**PROFESOR**

*Carlos De La Cruz Sosa*

**ALUMNOS**

*Fernández Guerrero Keb Sebastián*

*Ramírez Orozco Juan Carlos*

*Sánchez Herrera Armando Eduardo*

**TEMA**

Planes de Ejecución

Grupo 3TM3 Equipo 06 Tarea No. 03

**Introducción**

En la realización de la practica 02 de planes de ejecución hubo una consulta (Nombrada C en el reporte correspondiente), la cual causo dudas sobre como mejorar su ejecución. Hubo varias propuestas (puesta que cada equipo tenía una versión diferente o al menos esa era la idea), esta tarea trata de comparar la consulta realizada por el profesor implementando su versión usando ‘with’, ‘except’ y ‘not exists’ contra nuestra versión implementada en la práctica en cuestión.

Los resultados que esperaríamos es una tabla comparativa donde resalte el uso de cada consulta, así como un análisis de los porcentajes relativos de cada operación realizada en ella.

En las conclusiones se deberán destacar el conocimiento adquirido de las operaciones y como estas impactan al costo relativo final.

**Desarrollo**

La consulta que se realiza los análisis es la siguiente:

Listar los datos generales de las ordenes que tengan al menos los mismos productos de la orden con SalesOrderId = 43676

Para empezar, debemos analizar la estructura de la consulta realizada por nosotros y ver las diferencias sustanciales de cada una.

SELECT DISTINCT Salesorderid

FROM SalesOrderDetail AS OD

WHERE NOT EXISTS (SELECT \*

FROM (SELECT ProductID

FROM SalesOrderDetail

WHERE salesorderid = 43676) AS P

WHERE NOT EXISTS (SELECT \*

FROM SalesOrderDetail AS OD2

WHERE OD.SalesOrderID = OD2.SalesOrderID

AND OD2.ProductID = P.ProductID ))

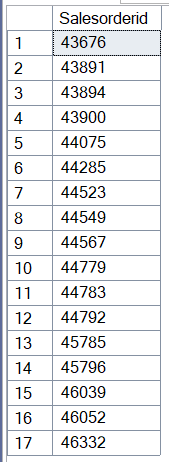


Ilustración Resultado de la consulta por los alumnos

En la sección final aquí se verifica si para cada SalesOrderID en la consulta externa (alias OD), existe un producto (ProductID) en la tabla SalesOrderDetail que no esté en la orden de venta 43676.

Si algún producto de SalesOrderID no existe en la orden 43676, la subconsulta retorna TRUE, y por lo tanto el NOT EXISTS de la consulta externa no se cumple, eliminando ese SalesOrderID de los resultados.

El problema mas grande que se logra ver es la necesidad de estar eliminando varias veces el conjunto de datos. Haciendo que la consulta tenga que usar índices muy específicos debido a la búsqueda de existencia.

La siguiente a analizar es la propuesta realizada por el profesor.

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

);

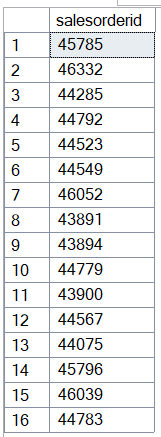


Ilustración Resultado de la consulta por el profesor

Para empezar esta solución cuenta con el uso de CTE (Common Table Expression) para poder hacer uso de solo dos consultas hacia la base de datos y que siempre se usen estas para todo el proceso, lo que ya con esto mejora bastante.

Esta consulta hace uso de un caso super funcional. Eliminar primero los resultados que por obviedad no pertenecen al resultado esperado. Lo logramos solo comparando la cantidad de productos que tiene cada cuenta. Son en total como 5 productos. Si resulta que el ticket que vemos no cuenta con esa cantidad lo descartamos de cuajo teniendo solo una tabla con datos coherentes que podemos analizar. Esto provoca que no se este continuamente consultando la base de datos y eliminando índice tras otro.

Una vez teniendo una tabla coherente con datos de rango, toca comparar y si es que esos datos coinciden con los otros tickets.

Esa es la gran diferencia. El uso del except que es un proceso que compara índices de dos tablas específicas. En contra del uso del not exists que va buscando índices que no estén en un rango especifico. A eso le añadimos el hecho de que la tabla no esta en orden genera problemas de múltiples búsquedas que van eliminando de a poco y provocando a la par que cada not exists anidado tenga que hacer búsquedas innecesarias para corroborar la información. Aun si usáramos ‘with’ no se vería tan reflejada como la implementación del profesor. Cuando se analicen los planes de ejecución tendrá que resaltar un proceso el cual contenga el mayor porcentaje de costo relativo.

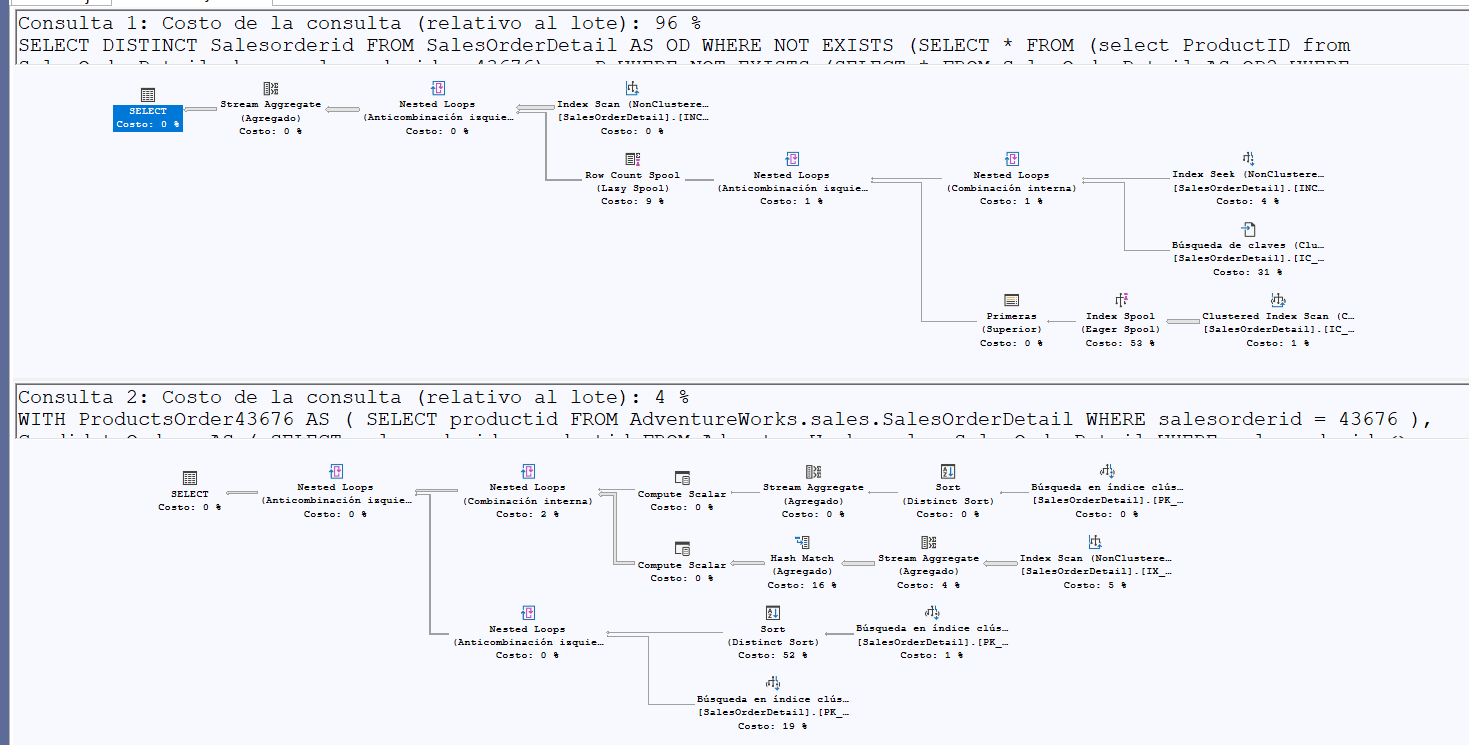


Ilustración Plan de ejecución de las consultas.

|  |  |  |
| --- | --- | --- |
| **Aspecto** | **Consulta 1 (NOT EXISTS)** | **Consulta 2 (WITH, EXCEPT)** |
| Uso de índices | Sí, pero repetido | Sí, más organizado |
| Estructura de bucles | Anidados (Nested Loops, Index Spool) | Hash y agregación (Hash Match) |
| Porcentaje de costo total | Alto (96% relativo) | Bajo (4% relativo) |
| Optimización | Baja, operaciones iterativas | Alta, procesamiento en lote |
| Legibilidad | Menor | Mayor |

**Conclusión**

En la primera consulta, destaca significativamente el uso del operador **Index Spool**, el cual crea un índice temporal que se reutiliza a lo largo de la ejecución de la consulta principal. Esta consulta está estructurada como una serie de filtros aplicados sucesivamente, lo que incrementa el costo de procesamiento. Cada cláusula NOT EXISTS genera una operación adicional sobre el índice temporal, reorganizando y depurando los datos, para posteriormente volver a insertar aquellos que no cumplen con las condiciones. Este proceso se repite con cada nueva cláusula NOT EXISTS, provocando una acumulación de operaciones que ralentizan considerablemente la ejecución.

En contraste, la segunda consulta, aunque persigue el mismo objetivo, define de forma más clara y estructurada el flujo de ejecución. En primer lugar, reduce el volumen de datos mediante una condición lógica con el operador >=, lo cual resulta computacionalmente eficiente. Posteriormente, emplea la combinación de EXCEPT con NOT EXISTS para realizar comparaciones directas entre conjuntos de datos ya ordenados, optimizando así el rendimiento. En esta consulta, la carga principal recae en el proceso de ordenamiento de datos, lo cual es considerablemente más eficiente que la depuración iterativa de registros utilizada en la primera consulta.