UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

FACULTAD DE CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA EN SISTEMAS

TEMA DE TESIS:

“CREACIÓN DE UN MODELO DE OPTIMIZACIÓN PARA LOS QUERY UTILIZANDO LA SENTENCIA SELECT DE SQL”

TESIS DE GRADO

PREVIO A LA OBTENCIÓN DEL TÍTULO DE

INGENIERA EN SISTEMAS

AUTORA:

MARÍA GABRIELA VARAS BELTRÁN

DIRECTOR DE TESIS:

ING. ARIOSTO VICUÑA, Msc

QUEVEDO – ECUADOR

2014

AUTORÍA Y CESIÓN DE DERECHOS

Yo, María Gabriela Varas Beltrán, declaro que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La universidad Técnica Estatal de Quevedo, puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

MARÍA GABRIELA VARAS BELTRÁN

AUTORA

CERTIFICACIÓN

El suscrito, Ing. Ariosto Vicuña, Docente de la Universidad Técnica Estatal de Quevedo, certifica que la Egresada María Gabriela Varas Beltrán, realizó la tesis de grado titulada **"Creación de un Modelo de Optimización para los Query utilizando la sentencia SELECT de SQL"**, bajo mi dirección, habiendo cumplido con las disposiciones reglamentarias establecidas para el efecto.

_____

DIRECTOR DE TESIS

UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

FACULTAD DE CIENCIAS DE LA INGENIERÍA

CARRERA DE INGENIERÍA EN SISTEMAS

Presentado al Consejo Directivo como requisito previo a la obtención del título de Ingeniera en Sistemas.

Aprobado:

_____

Dr. PhD. Amilkar Puris Cáceres, Ing
PRESIDENTE DEL TRIBUNAL DE TESIS

_____ _____

Ing. Washington Chiriboga
MIEMBRO DEL TRIBUNAL DE TESIS

Ing. Carlos Márquez de la Plata
MIEMBRO DEL TRIBUNAL DE TESIS

QUEVEDO – LOS RÍOS – ECUADOR

AÑO 2014

DEDICATORIA

A Dios quien me dio la vida y supo guiarme por el buen camino.

Con mucho cariño, a mis padres César Varas y Julieta Beltrán por el apoyo incondicional que me han brindado en mi etapa de estudio y por darme ejemplos dignos de superación.

A mi abuela Alejita por su amor sincero, por incentivarme a que sea una persona de éxito.

A mi hermano con quien he tenido el privilegio de compartir mi etapa estudiantil, juntos aprendimos a vivir y somos amigos de toda la vida.

A mis familiares, amigos, a mi gran amor Jim por ser mi compañero inseparable, por sus sonrisas de ánimo y su amor que me brinda.

AGRADECIMIENTO

En primer lugar a Dios por todas la bendiciones que me ha dado en la vida y por brindarme la sabiduría para lograr alcanzar esta meta.

A mis extraordinarios padres por sus consejos, cariño, entrega, motivación, por sus valores inculcados han guiado mis pasos por el camino del bien.

A mi tutor de tesis por incentivarme a la investigación y por su apoyo brindado durante la realización de este trabajo.

A mi tía Petita un agradecimiento especial y sincero por su colaboración a lo largo de mi carrera y por haber estado pendiente de que cumpla este objetivo de mi vida.

A la hermana Ruth Christhian por darme valiosos consejos y por su apoyo brindado.

A mis estimados maestros por sus sabias enseñanzas, los conocimientos que me brindaron serán la base de mi vida profesional.

(DUBLIN CORE) ESQUEMA DE CODIFICACIÓN			
1	**Título/Title**	M	“Creación de un modelo de optimización para los query utilizando la sentencia SELECT de SQL”
2	**Creador/Creator**	M	Varas Beltrán María Gabriela
3	**Materia/Subject**	M	Ciencias de la Ingeniería; Carrera Ingeniería en Sistemas; Bases de datos; Optimización de query.
4	**Descripción/Description**	M	La presente investigación pretende ser una guía para la construcción de query eficientes con millones de registros. El objetivo consiste en construir un modelo para la optimización de consultas realizadas mediante el SQL.
5	**Editor/Publisher**	M	FCI: Carrera Ingeniería en Sistemas; Varas Beltrán María Gabriela
6	**Colaborador/Contributor**	O	Ninguno
7	**Fecha/Date**	M	21/03/2014
8	**Tipo/Type**	M	Tesis de Grado
9	**Formato/Format**	R	Microsoft Office Word 2013 (.docx) Adobe Acrobat Document (.pdf)

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICOS

RESUMEN

La presente tesis tiene como objetivo la construcción de un modelo de optimización para las Query utilizando la sentencia SELECT de SQL.

Para este propósito se utilizó un segmento de la base de datos de OpenBravo POS, dicho segmento contenía tablas con el mayor número de relaciones. A partir de estas tablas se crearon scripts para generar los 10 y 20 millones de registros; para crear el modelo de optimización se establecieron cuatro requerimientos de consultas cada una de estas se escribieron de ocho formas diferentes: select simple, utilizando subconsultas, usando joins, combinando subquery con join y por último aplicándole índice a cada una de estas. Las pruebas se realizaron en los gestores de base de datos PostgreSQL 9.2, SQL Server 2008 R2 Developer y Oracle 11g Enterprise Edition.

Los resultados demostraron que el modelo de optimización mejora el rendimiento de las query; los tiempos de respuesta y utilización de recursos en cada DBMS son distintos debido a que cada uno de estos tienen un optimizador interno el cual determina la forma más eficiente de procesar la consulta, lo podemos visualizar a través de los planes de ejecución.

ABSTRACT

The following thesis has the objective for building the optimization model for the Query using the SQL SELECT statement.

To this end I utilized a segment of the database OpenBravo POS, the segment tables containing the largest number of relations are used. From these tables were created scripts to generate 10 to 20 million records, to create the model of query optimization four requirements each of these were written in eight different forms were established: simply select, using subqueries, using joins, combining subquery to join index and finally applying to each of these. The tests were performed in databases managers PostgreSQL 9.2, SQL Server 2008 R2 Developer and Oracle 11g Enterprise Edition.

The results showed that the optimization model improves the performance of the query, response times and resource utilization in each DBMS are different because each one of these I have an internal optimizer which determines the most efficient way to process the query you can visualize through the execution plans.

CAPITULO I

MARCO CONTEXTUAL DE LA INVESTIGACIÓN

1.1 INTRODUCCIÓN

Las consultas a la base de datos resulta uno de los mayores problemas de degradación de los sistemas de información cuando se llenan los registros de las base de datos. Si el sistema realiza operaciones masivas de inserción de registros debemos preocuparnos de la sentencia SELECT.

Lo usual es que al principio se trabaja con tablas que no contienen demasiados registros, seguramente jamás nos demos cuenta que nuestra consulta SQL no fue desarrollada de la mejor forma porque los tiempos de respuesta son aceptables, pero cuando la base de datos comienza a crecer, suele suceder que muchos procesos o reportes del sistema se vuelven extremadamente lentos (Alfaro, 2011).

Esto significa que algunas de las consultas SQL que hemos escrito no estaban preparadas para trabajar con grandes volúmenes de datos. También puede suceder que el problema no es únicamente de la forma en que desarrollamos una consulta, sino también que están mal establecidos algunos índices de las tablas u otros temas de configuración.

Por esto se debe aplicar la optimización a las consultas que resultan costosas o que se ejecutan con mayor frecuencia para mejorar en gran medida el rendimiento de la base de datos y garantizar la ganancia de muchos segundos o minutos que el usuario debe esperar al momento de realizar consultas desde alguna aplicación.

La presente investigación está orientada hacia la CONSTRUCCIÓN DE UN MODELO PARA LA OPTIMIZACIÓN DE LA SENTENCIA SELECT, y así establecer consultas eficientes que al ser ejecutadas en diferentes motores de bases de datos tendrá un rendimiento óptimo.

1.2 JUSTIFICACIÓN

Esta investigación es una continuación de lo investigado en la tesis del Ingeniero Carlos Hidalgo donde trata sobre la importancia de optimizar las consultas. La finalidad de esta investigación es crear un modelo de optimización de la sentencia SELECT que permita mejorar el rendimiento de las Query, porque de estas depende gran parte del rendimiento de la base de datos.

Es importante tener las Query optimizadas porque reducirán la cantidad de trabajo extra que realizan los motores de base de datos, disminuirán recursos y aumentarán la eficiencia, logrando así mejores resultados en cada consulta que se realice a la base de datos.

Esta investigación servirá de guía a los estudiantes de la Facultad de Ciencias de la Ingeniería, Carrera Ingeniería en Sistemas que estén incursionando en el entorno de las bases de datos, para que obtengan un conocimiento oportuno sobre la construcción de Query eficientes sobre millones de datos.

1.3 PROBLEMATIZACIÓN

A medida que los registros de la base de datos aumentan las Query se vuelven lentas y comienzan a surgir problemas en cuanto a rendimiento. Una de las principales causas es su mala formulación, generalmente una consulta puede expresarse de distintas formas cada una de estas indica una estrategia para lograr el mejor rendimiento; de ahí que algunas pueden ser más óptimas que otras.

Otra de las razones por las cuales se produce esta problemática es que no se presta mayor atención a las consultas de selección siendo estas las responsables de la mayor pérdida de tiempo debido a su falta o pobre optimización. En este grupo se encuentra la sentencia SELECT que provee de una serie de cláusulas que permiten seleccionar las tablas, filtrar la

información, agrupar los datos, ordenar el resultado, unir consultas. Se la considerada la más potente y compleja de todas las sentencias SQL porque este tipo de operaciones donde notaremos como, poco a poco se va degradando un sistema.

Por otro lado están las consultas de acción que engloba las sentencias INSERT, UPDATE y DELETE comparadas con la complejidad que posee la sentencia SELECT estas resultan más simples

Otro factor que afecta el rendimiento de las Query es el uso inapropiado de índices (INDEX) o su falta de uso.

1.3.1 FORMULACIÓN DEL PROBLEMA.

¿La forma de escribir una consulta y la utilización de índices mejora el rendimiento de las mismas?

1.4 OBJETIVOS

1.4.1 GENERAL

Construir un modelo para la optimización de consultas utilizando la sentencia SELECT realizadas mediante el Statement Query Language (SQL).

1.4.2 ESPECÍFICOS

- Determinar la técnica más eficaz para consultar datos de varias tablas a través de los joins.
- Establecer la mejor estrategia de búsqueda con el uso de subconsultas.
- Evaluar el beneficio que aportan los índices en la recuperación de información.

1.5 HIPÓTESIS

1.5.1 GENERAL

Hi: La aplicación de un modelo de optimización de consultas (Query) mejora el desempeño de las Query.

1.5.2 OPERACIONAL

Hi1: El número de registros y el tipo de DBMS en los que se ejecutan las query optimizadas producen tiempos de respuestas diferentes.

1.5.3 VARIABLES

1.5.3.1 VARIABLE INDEPENDIENTE

Tabla 1. Matriz de Conceptualización

Matriz de Conceptualización y Operacionalización de Variable			
Variable	Definición	Dimensiones	Indicadores
Modelo de Optimización de Query	Guía para construir consultas eficaces	Consulta SQL	- Índices - Sentencias con Subconsultas - Sentencias con Join

1.5.3.2 VARIABLE DEPENDIENTE

Tabla 1. Matriz de Conceptualización

Matriz de Conceptualización y Operacionalización de Variable			
Variable	Definición	Dimensiones	Indicadores
Desempeño de las Query	Comportamiento de las Query	Eficiencia	Tiempo de Respuesta
		Utilización de Recursos	Evaluación del plan de ejecución

CAPITULO II

MARCO TEÓRICO

2.1 FUNDAMENTACIÓN TEÓRICA

2.1.1 Componentes de la sentencia SELECT

2.1.1.1 Cláusulas.

Las cláusulas son condiciones de modificación para definir los datos que desea seleccionar o manipular.

Tabla 2. Cláusulas

Cláusula	Descripción
FROM	Especifica la tabla de la cual se van a seleccionar los registros.
WHERE	Especifica las condiciones que deben reunir los registros que se van a seleccionar.
GROUP BY	Separar los registros en grupos específicos.
HAVING	Expresa la condición que debe satisfacer cada grupo.
ORDER BY	Ordena los registros seleccionados de acuerdo con un orden específico.

2.1.1.2 Operadores Lógicos.

Tabla 3. Operadores Lógicos

Operador	Uso
AND	Es el “y” lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el “o” lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión (Garavito J. , 2007).

2.1.1.3 Operadores de Comparación.

Tabla 4. Operadores de Comparación

Operador	Descripción
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor o igual que
>=	Mayor o igual que
=	Igual que
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en la comparación de un modelo.
IN	Utilizado para especificar registros de una base de datos.

2.1.1.4 Funciones de Agregado.

Tabla 5. Funciones de Agregado

Función	Descripción
AVG	Utilizada para calcular el promedio de los valores de un campo determinado.
COUNT	Utilizada para devolver el número de registros de selección.
SUM	Utilizada para devolver la suma de todos los valores de un campo determinado.
MAX	Utilizada para devolver el valor más alto de un campo especificado.
MIN	Utilizada para devolver el valor más bajo de un campo especificado (Casares, S.f).

2.1.2 Consultas de Selección

2.1.2.1 Consultas básicas.

Una consulta SQL básica puede constar con un máximo de seis cláusulas, de las cuales sólo dos son obligatorias (SELECT y FROM) (Casares, S.f).

2.1.2.2 Consultas con predicados.

El predicado se incluye entre la cláusula y el primer nombre del campo a recuperar, los posibles predicados son:

Tabla 6. Predicados

Predicados	Descripción
ALL o (*)	Devuelve todos los campos de la tabla
TOP	(Casares, S.f) Devuelve un determinado número de registros de la tabla
DISTINCT	Omite los registros que contienen datos duplicados en los campos seleccionados
DISTINCT ROW	Omite los registros duplicados basándose en la totalidad del registro y no sólo en los campos seleccionados (Escarrega).

2.1.2.3 Alias.

Hay dos tipos de alias que se utilizan con mayor frecuencia:
Alias de columna: Existen para ayudar en la organización de resultado.
Alias de tabla: Es conveniente cuando desea obtener información de dos tablas separadas.

2.1.3 Consultas de Acción

Las consultas de acción son aquellas que no devuelven ningún registro, son las encargadas de acciones como añadir, borrar y modificar registros.

2.1.3.1 Delete.

Crea una consulta de eliminación que elimina los registros de una o más de las tablas listadas en la cláusula FROM que satisfagan la cláusula WHERE. Esta consulta elimina los registros completos, no es posible eliminar el contenido de algún campo en concreto. Su sintaxis es:

DELETE Tabla.* FROM Tabla WHERE criterio DELETE es especialmente útil cuando se desea eliminar varios registros. En una instrucción DELETE con múltiples tablas, debe incluir el nombre de tabla (Tabla.*). Si especifica más de una tabla desde la que eliminar registros, todas deben ser tablas de muchos a uno. Si desea eliminar todos los registros de una tabla, eliminar la propia tabla es más eficiente que ejecutar una consulta de borrado (Escarrega).

2.1.3.2 Insert Into.

Agrega un registro en una tabla. Se la conoce como una consulta de datos añadidos. Se puede utilizar la instrucción INSERT INTO para agregar un registro único a una tabla, utilizando la sintaxis de la consulta de adición de registro único. En este caso, su código especifica el nombre y el valor de cada campo del registro. También se puede utilizar INSERT INTO para agregar un conjunto de registros pertenecientes a otra tabla o consulta utilizando la cláusula SELECT ... FROM. En este caso la cláusula SELECT especifica los campos que se van a agregar en la tabla destino especificada. La tabla destino u origen puede especificar una tabla o una consulta. Si la tabla destino contiene una clave principal, hay que asegurarse que es única, y con valores no-Null ; si no es así, no se agregarán los registros.

Se pueden averiguar los registros que se agregarán en la consulta ejecutando primero una consulta de selección que utilice el mismo criterio de selección y ver el resultado. Una consulta de adición copia los registros de una o más tablas en otra. Las tablas que contienen los registros que se van a agregar no se verán afectadas por la consulta de adición. En lugar de agregar registros

existentes en otra tabla, se puede especificar los valores de cada campo en un nuevo registro utilizando la cláusula VALUES. Si se omite la lista de campos, la cláusula VALUES debe incluir un valor para cada campo de la tabla, de otra forma fallará INSERT.

2.1.3.3 Update.

Crea una consulta de actualización que cambia los valores de los campos de una tabla especificada basándose en un criterio específico.

Update es especialmente útil cuando se desea cambiar un gran número de registros o cuando éstos se encuentran en múltiples tablas. Puede cambiar varios campos a la vez. Para saber qué registros se van a cambiar, hay que examinar primero el resultado de una consulta de selección que utilice el mismo criterio y después ejecutar la consulta de actualización. Si en una consulta de actualización suprimimos la cláusula WHERE todos los registros de la tabla señalada serán actualizados.

2.1.4 Subconsultas

Una subconsulta es una consulta SELECT que está anidada en una instrucción SELECT, INSERT, UPDATE o DELETE, o dentro de otra subconsulta. En una subconsulta, la instrucción SELECT nos proporciona un conjunto de uno o más valores que se utilizan para evaluar una expresión (Alcalde, 2013).

Una subconsulta se puede utilizar en cualquier parte en la que se permita una expresión. La consulta SELECT de una subconsulta se incluye siempre entre paréntesis. Se suele utilizar una subconsulta en lugar de una expresión en la lista de campos de una instrucción SELECT, igualmente una subconsulta puede anidarse dentro de la cláusula WHERE o HAVING.

Se puede disponer de hasta 32 niveles de anidamiento, aunque el límite varía dependiendo de la memoria disponible y de la complejidad del resto de las expresiones de la consulta.

2.1.5 Consultas Combinadas

Habitualmente cuando necesitamos recuperar la información de una base de datos nos encontramos con que esta se encuentra repartida en varias tablas, referenciadas a través de un campo llave. Está forma de almacenar la información no resulta muy útil a la hora de consultar los datos. SQL proporciona una forma fácil de mostrar los datos repartidos en varias tablas, las consultas combinadas o JOINS (Herrarte, 2005).

2.1.5.1 Tipos de consultas combinadas

2.1.5.1.1 Combinación interna.

La combinación interna es excluyente. Se realiza mediante la cláusula INNER JOIN que combina registros de dos tablas siempre que haya concordancia de valores en un campo común.

2.1.5.1.2 Combinación Externa.

No es excluyente. Puede ser LEFT OUTER JOIN con el cual obtenemos todos los registros de la tabla que situemos a la izquierda, o RIGHT OUTER JOIN que adquiere el efecto contrario.

2.1.6 Índices

Un índice es una estructura de datos que permite acceder a diferentes filas de una misma tabla a través de un campo (o campos clave). Permite un acceso mucho más rápido a los datos (Herrarte, 2005).

Permiten mejorar el rendimiento de las consultas a la base de datos. Ayudan a mejorar el procesamiento de las sentencias SELECT pero también de las sentencias UPDATE y DELETE. Cuando es necesario recuperar los registros de una tabla proveen el path de acceso más eficiente, además ayudan a forzar la integridad referencial en la base de datos (Primary Key, Foreing Key, Unique Columns) (Pullas, 2006).

2.1.6.1 Uso eficiente de índices.

Los DBMS pueden acceder a las tablas de dos modos:

- Secuencialmente (FULL SCAN).
- Usando índices.

El diseñador crea los índices en función del uso más frecuente de los datos, pero es el programador el que decide, en último término, si en la sentencia en que está trabajando hará que el gestor de base de datos emplee los índices o haga un acceso secuencial. En general, el acceso por índice es preferible. Sin embargo, si se prevé que un alto porcentaje de los datos será consultado (digamos más de un 25%) un acceso secuencial puede ser más rápido.

2.1.6.2 Condiciones para el uso de índices.

Los gestores de base de datos podrán usar un índice sólo si se cumplen estas dos condiciones:

- La columna es referenciada en un predicado.
- La columna indexada no está modificada por ninguna función u operador aritmético.

Que se cumplan ambas condiciones no implica necesariamente que el índice sea usado. El optimizador decidirá si es apropiado o no emplearlo.

2.1.6.3 Índices y valores nulos.

Siempre que sea posible, es preferible definir las columnas como NOT NULL

(Incluso buscando un valor como cero o espacio para sustituir el significado del nulo). Esto permite el uso de índices en un mayor número de sentencias.
En efecto, ORACLE principalmente no emplearán índices si el predicado contiene las cláusulas: IS NULL o IS NOT NULL, esto se debe a que el gestor no almacena los valores nulos en el índice. En general un índice no contendrá columnas que admitan valores nulos. Cuando la búsqueda se realiza sobre uno de estos índices, aunque el predicado sea de igualdad e identifique completamente la fila deseada, el gestor accede parcialmente en modo secuencial (RANGE SCAN).

Aun cuando se haya definido un índice sobre una columna que admite nulos, es posible, en algunos casos, obligar al optimizador a usar parcialmente los índices.

2.1.6.4 Reglas en el diseño de índices.

La mejor solución NO es crear índices para cada una de las columnas de la cláusula WHERE. More is not better!.

Su meta a nivel de optimización de índices debería utilizar solo aquellos que son necesarios, para esto, el mejor punto de partida es realizar una búsqueda en su base de datos de los índices que utilizan la misma columna de una tabla y adicionalmente de aquellos índices que utilizan el mismo par de columnas de una tabla, esto, con la finalidad de eliminar redundancia.

Cualquier tabla con un Primary Key tiene ya un índice y no es necesario crear un índice adicional.

Los índices multi-columna (más de 3 columnas) en muchos casos pueden no proveer un acceso tan eficiente a los datos como un índice de dos columnas.
No tenga temor de utilizar índices temporales, si va a correr un reporte en las noches y necesita un índice específico créelo y bórrelo luego cuando haya terminado.

Evite indexar tablas muy pequeñas porque normalmente es más eficaz realizar un examen de tablas (Pullas, 2006).

2.1.6.5 Índices en Postgresql 9.2

2.1.6.5.1 B-Tree.

Es el valor predeterminado que se obtiene cuando usted crea el índice. El B es sinónimo de equilibrado, y la idea es que la cantidad de datos en ambos lados del árbol es más o menos el mismo. Por lo tanto el número de niveles que hay que atravesar para encontrar filas es siempre en el mismo. Índices B-Tree se pueden utilizar para la igualdad y la gama consultas de manera eficiente. Pueden operar en contra numérico, texto o valores NULL (Heroku dev center, 2014).

2.1.6.5.2 Índices hash.

Sólo son útiles para comparaciones de igualdad, pero casi nunca son usados, ya que no son transacciones seguras, por lo que la ventaja sobre el uso de un B-Tree es bastante pequeño.

2.1.6.5.3 Índices parciales.

Un índice parcial cubre sólo un subconjunto de datos de una tabla. Se trata de un índice con una cláusula WHERE. La idea es aumentar la eficiencia del índice mediante la reducción de su tamaño. Un índice más pequeño tiene menos capacidad de almacenamiento, es más fácil de mantener, y es más rápido para escanear.

2.1.6.5.4 Índices únicos.

Un índice único garantiza que la tabla no tendrá más de una fila con el mismo valor. Es ventajoso para crear índices únicos por dos razones: la integridad y el

rendimiento de datos. Búsquedas en un índice único son generalmente muy rápido.

2.1.6.5.5 Índices Ordenados.

Entradas de índice de árbol B se clasifican en orden ascendente por defecto. En algunos casos tiene sentido para suministrar un orden diferente para un índice.

2.1.6.6 Índices en SQL Server Developer 2008 R2

2.1.6.6.1 Índices agrupados (Clustered Indexes).

Un índice agrupado almacena las filas de datos reales en el nivel hoja del índice. Una característica importante del índice agrupado es que los valores indexados se ordenan de forma ascendente o descendente. Como resultado, sólo puede haber un índice agrupado en una tabla o vista. Además, los datos de una tabla se ordenan sólo si un índice agrupado se ha definido en una tabla (Sheldon, 2008).

La tabla tiene un índice agrupado se conoce como tabla agrupada. Una tabla que no tiene un índice agrupado se conoce como un montón.

2.1.6.6.2 Los índices no agrupados (Nonclustered Indexes).

A diferencia de un clúster en un índice, los nodos hoja de un índice no agrupado contienen sólo los valores de las columnas indizadas y localizadores de filas que apuntan a las filas de datos reales, y no contienen las filas de datos. Esto significa que el motor de consulta debe dar un paso más con el fin de localizar los datos reales.

Estructura una fila de localizador depende de si se apunta a una tabla agrupada o en un montón. Si se hace referencia a una tabla agrupada, los

puntos de localización de fila para el índice agrupado, utilizando el valor del índice agrupado para desplazarse a la fila de datos correcta. Si hace referencia a un montón, el localizador de fila apunta a la fila de datos actual.
Los índices no agrupados no pueden ser clasificados como índices agrupados, sin embargo, puede crear más de un índice agrupado por tabla o vista. SQL Server 2005 admite un máximo de 249 índices no agrupados y SQL Server 2008 admiten hasta 999. Esto ciertamente no significa que usted debe crear que muchos índices. Los índices pueden tanto ayudar y obstaculizar el rendimiento.

Además de ser capaz de crear varios índices no agrupados en una tabla o vista, también puede agregar columnas incluidas en el índice. Esto significa que usted puede almacenar en el nivel de hoja no sólo los valores de la columna indexada, sino también los valores de las columnas no indexadas. Esta estrategia le permite obtener alrededor de algunas de las limitaciones de los índices. Por ejemplo, puede incluir columnas no indexadas con el fin de superar el límite de tamaño de las columnas indizadas (900 bytes en la mayoría de los casos).

2.1.6.6.3 Tipos de indexación.

Además de un índice está agrupado o no agrupado, que puede ser configurado de otras maneras:

2.1.6.6.4 Índice compuesto.

Un índice que contiene más de una columna. Tanto en SQL Server 2005 y 2008, se puede incluir un máximo de 16 columnas en un índice, siempre que el índice no supera el límite de 900 bytes. Ambos índices agrupados y no agrupados pueden ser índices compuestos.

2.1.6.6.5 Índice Único.

Un índice que garantiza la singularidad de cada valor de la columna indexada. Si el índice es un compuesto, la singularidad se aplica a través de las columnas como un todo, no en las columnas individuales. Por ejemplo, si desea crear un índice en las columnas Nombre y Apellidos de una tabla, los nombres juntos deben ser únicos, pero los nombres individuales se pueden duplicar.

Un índice único se crea automáticamente cuando se define una restricción de clave única o primaria:

- **Clave principal:** Cuando se define una restricción de clave principal en una o más columnas, SQL Server crea automáticamente un índice agrupado único caso de un índice agrupado no existe en la tabla o la vista. Sin embargo, puede reemplazar el comportamiento predeterminado y definir un índice único no agrupado en la clave principal.

- **Único:** Cuando se define una restricción de unicidad, SQL Server crea automáticamente un índice no agrupado único. Puede especificar que se crea un índice agrupado único si un índice agrupado no existe ya en la tabla.

- **Cubriendo índice:** Un tipo de índice que incluye todas las columnas que se necesitan para procesar una consulta particular. Por ejemplo, la consulta podría recuperar las columnas Nombre y Apellidos de una tabla, sobre la base de un valor en la columna ContactID. Usted puede crear un índice de cobertura que incluye las tres columnas.

2.1.6.7 Índices en Oracle 11g

2.1.6.7.1 Lectura/Escritura.

2.1.6.7.1.1 B-tree (árboles binarios).

El índice B-Tree es el tipo de índice más común en una base de datos Oracle. Es el índice default, es decir que si uno crea un índice sin especificar el tipo, Oracle lo creará de tipo B-Tree.

Se estructura como un árbol cuya raíz contiene múltiples entradas y valores de claves que apuntan al siguiente nivel del árbol (García, 2008).

2.1.6.7.2 Sólo lectura (read only)

2.1.6.7.2.1 Bitmap.

A diferencia de los índices B-Tree, los índices de tipo Bitmap utilizado por ORACLE, usa una fracción de espacio mucho menor representando los rowids como valores binarios 0 o 1. Los índices Bitmap son aconsejables en situaciones en que los diferentes valores que puede tomar la columna son relativamente pocos. Ejemplos: sexo, estado civil, entre otros. Cuantos menos valores posibles, mejor. A medida que crece la cantidad de valores posibles, aumentará el tamaño del índice. Es por ello que las búsquedas por índices son una alternativa a recorrer la tabla completa al intentar encontrar un valor, es decir que es posible reducir los I/O.

2.1.6.7.4 Índices creados por Oracle de manera automática.

Un índice UNIQUE basado en B*-tree para mantener las columnas que se hayan definido como clave primaria de una tabla utilizando el constraint PRIMARY KEY de una tabla no organizada por índice.

Un índice UNIQUE basado en B*-tree para mantener la restricción de unicidad de cada grupo de columnas que se haya declarado como único utilizando el constraint UNIQUE.

Un índice basado en B*-tree para mantener las columnas que se hayan definido como clave primaria y todas las filas de una tabla organizada por índice.

Un índice basado en hashing para mantener las filas de un grupo de tablas (“cluster”) organizado por hash (Paredes, 2011).

2.1.7 Optimización de Sentencias.

Algunos casos típicos de situaciones que incitan a optimizar una consulta son los siguientes:

1. Muchos optimizadores de consultas no utilizan los índices en presencia de expresiones aritméticas (por ejemplo, Sueldo/360>10.50), comparaciones numéricas de atributos de tamaños y precisiones diferentes (por ejemplo a.id = b.id donde a.id es del tipo INTEGER y b.id es del tipo SMALLINTEGER), comparaciones con NULL (por ejemplo, Fecha is NULL), y comparaciones de subcadenas (por ejemplo, Apellido LIKE perez ‘%ez’).

2. No se utilizan con frecuencia los índices en las consultas anidadas con IN; por ejemplo la siguiente consulta:

   ```
   SELECT  Dni FROM EMPLEADO
   WHERE Dno IN (SELECT numDpto FROM DEPARTAMENTO
                 WHERE dniDirector=’333445555’);
   ```

   no puede utilizar el índice por Dno en EMPLEADO, mientras que el uso de Dno=numDpto en la cláusula WHERE con la consulta de un solo bloque puede provocar que se utilice en índice.

3. Algunos DISTINCTS pueden ser redundantes y puede evitarse sin tener que cambiar el resultado DISTINCT a menudo provoca una operación de ordenación y debe evitarse siempre que sea posible.

4. El uso innecesario de tablas de resultado temporales pueden evitarse colapsando varias consultas en una sola a menos que se necesite la relación temporal para algún procesamiento intermedio.

5. En algunas situaciones que implican el uso de consultas correlativas, los temporales son útiles.
   Considere la siguiente consulta:

```
SELECT dni
FROM EMPLEADO E
WHERE Sueldo=SELECT MAX (Sueldo)
                FROM EMPLEADO AS M
               WHERE M.Dno=E.Dno;
```

   Esto tiene el peligro potencial de buscar en toda la tabla EMPLEADO M interior por cada tuple de la table EMPLEADO E exterior. Para que esto sea más eficaz, se puede dividir en dos consultas, de modo que la primera calcula el sueldo máximo de cada departamento:

```
SELECT MAX (Sueldo) as SueldoMax, Dno INTO TEMP
FROM EMPLEADO
GROUP BY Dno;

SELECT Dni
FROM EMPLEADO, TEMP
WHERE Sueldo=SueldoMax AND EMPLEADO.Dno=TEMP.Dno;
```

6. Si son posibles varias opciones para una condición de concatenación, elija una que utilice un índice agrupado y evite las que contienen comparaciones de cadena. Por ejemplo, asumiendo que el atributo NOMBRE es una clave candidata en EMPLEADO Y ESTUDIANTE, es

mejor utilizar EMPLEADO.Dni = ESTUDIANTE.Dni como condición de concatenación que EMPLEADO.Nombre = ESTUDIANTE.Nombre si Dni tiene un índice agrupado en una o en las dos tablas.

7. Una rareza de los optimizadores de consultas es que el orden de las tablas en la cláusula FROM puede afectar el procesamiento de la concatenación. Si el caso, es posible tener que cambiar este orden para que se explore la más pequeña de las dos relaciones y se utilice la más grande con un índice adecuado.

8. Algunos optimizadores de consultas funcionan peor con las consultas anidadas que con las no anidadas.

2.1.7.1 Optimizando el uso del operador NOT.

Siempre que aparece la negación de una igualdad (!= o NOT=), ORACLE no emplea índices. Así, aunque un índice se haya definido para la columna X, la sentencia:

```
SELECT X,Y,Z
FROM TAB
WHERE X != 0
```

no empleará dicho índice. Este comportamiento se justifica porque es de suponer que la consulta anterior recupere la mayor parte de las filas de la tabla; en estos casos, suele ser más rápida una búsqueda secuencial.

En la situación anterior, si además suponemos que se ha definido un índice sobre la columna Y, la sentencia:

```
SELECT X,Y,Z
FROM TAB
WHERE X != 0
AND Y = ‘A’
```

podría emplear el índice sobre la columna Y a pesar de la negación sobre la columna X.

2.1.7.2 Optimizando el uso del operador OR.

Las sentencias que incluyen el operador OR pueden usar índices si estos se han definido sobre todas las columnas que intervienen. Si alguna de las columnas no está indexada, es posible que el optimizador decida no emplear índices. Esta última circunstancia queda fuera del control del programador (suele darse en sentencias que incluyen joins externos).
Cuando todas las columnas están indexadas, ORACLE puede usar los índices eficientemente. Por ejemplo, si se han definido índices sobre las columnas X e Y, la sentencia:

```
SELECT X,Y,Z
FROM TAB
WHERE X = 1
OR Y = 'A'
```

transformada en algo similar a la unión de las dos consultas siguientes:

```
SELECT X,Y,Z
FROM TAB
WHERE X = 1
```

```
SELECT X,Y,Z
FROM TAB
WHERE Y = 'A'
AND X != 1
```

Conociendo este comportamiento, es mejor situar al principio la condición más selectiva.
La optimización sobre los OR's se aplica, en igual medida, a la cláusula IN, dado que el predicado:

```
WHERE Y IN (A,B,C)
```

es equivalente a:

WHERE Y = ‘A’
OR Y = ‘B’
OR Y = ‘C’

2.1.7.3 Optimizando el uso del operador AND.

Cuando en un predicado intervienen varias condiciones unida por el operador AND, puede obtenerse un ligero aumento de rendimiento situando la condición más selectiva al principio.

Esto se debe al orden en que el parser de ORACLE analiza las sentencias: las condiciones unidas por el operador AND dejan de evaluarse cuando cualquiera de ellas no es cierta.

2.1.7.4 Ordenación.

El gestor de ORACLE dispone de unas rutinas de ordenación y combinación de datos (SORT/MERGE) altamente especializadas que optimizan el rendimiento de aquellas operaciones donde se requiere una ordenación previa: CREATE INDEX, SELECT DISTINCT, ORDER BY, GROUP BY y ciertos tipos de joins.
De forma simplificada podemos decir que el gestor ordena los datos en pequeños grupos (runs) los cuales combina (merge) para obtener un resultado final. Existen varios parámetros que permiten al DBA influir sobre las rutinas de ordenación de ORACLE de forma que un alto porcentaje (más del 98%) de las ordenaciones se realicen en memoria.
Es importante conocer que la única forma de garantizar el orden de recuperación de un conjunto de datos es con la cláusula ORDER BY. Según la definición de índices asociada a las tablas consultadas puede obtenerse un orden adecuado sin especificar la cláusula ORDER BY.

Cuando se realiza una consulta que contiene un GROUP BY, el gestor debe ordenar los datos antes de agruparlos, por lo que es mucho más eficiente imponer todas los criterios de selección como un predicado WHERE (antes de

la ordenación) que con la cláusula HAVING. Esta última debe reservarse para cuando las condiciones se desean imponer sobre criterios de grupo. Por ejemplo, la consulta:

```
SELECT Y, AVG(X)
FROM TAB
WHERE Y LIKE 'A%'
GROUP BY Y
```

es preferible a:

```
SELECT Y, AVG(X)
FROM TAB
GROUP BY Y
HAVING Y LIKE 'A%'
```

2.1.8 Optimizando joins.

El empleo de índices en los joins resulta crítico para el rendimiento global de toda la aplicación. Siempre que se necesite utilizar un join, el programador debe cuidar la forma de codificarlo de modo que se aprovechen eficientemente los índices.

La regla fundamental consiste simplemente en asegurar que las columnas claves del join, aquellas que relacionan los datos de cada par de tablas, estén indexadas. Además, debe escribirse la sentencia de forma que las funciones y operadores no anulen el uso de índices.

Para realizar la consulta generada por un join, el gestor elige una de las tablas como conductora (driver) del join. ORACLE lee cada fila de esta tabla que cumpla las condiciones dadas y, para cada una de estas filas busca en las demás tablas las filas que cumplan las condiciones adicionales especificadas en el join.

El acceso a la tabla conductora es siempre secuencial. Por el contrario, el acceso al resto de tablas que participan en el join puede realizarse por medio de índices, siempre que esto sea posible.

Así pues, el objetivo del programador será conseguir que la tabla conductora sea aquella que presente, a priori, un camino de acceso más lento (sin índices, índices no únicos o con menor número de filas).

El optimizador de ORACLE selecciona la tabla conductora en función de los índices definidos y de sus propias reglas de optimización.

2.1.8.1 Joins no indexados.

Los joins implican siempre una condición de igualdad (equijoins) sobre uno o varios campos de las tablas que participan en el mismo. A dichos campos, que relacionan las filas recuperadas de las diferentes tablas, le llamaremos campos clave del join.

Siempre en un join el optimizador intentará utilizar los índices definidos sobre los campos clave del mismo. Si no es posible usar índices (por ejemplo, porque no existan), ORACLE debe realizar una ordenación de todas las tablas y una combinación posterior de las mismas (sort-merge join).

Concretamente, el gestor seleccionará los datos de cada tabla aplicando las condiciones dadas para los campos que no son claves del join. Los subconjuntos obtenidos a partir de cada tabla, son ordenados por separado. Los resultados se combinan en una nueva relación basándose en las condiciones clave del join.

Las condiciones adicionales más selectivas (aquellas que no son claves) deben situarse al final del predicado. Esto disminuirá el número de comprobaciones durante la primera fase de la operación anteriormente descrita. En un join no indexado, por tanto, ninguna de las tablas es la conductora.

2.1.8.2 Joins indexados.

Este tipo de consulta obliga al optimizador a realizar un complejo trabajo para decidir la tabla conductora. En el join de dos tablas, si sólo existen índices usables para una de ellas, el optimizador elegirá la otra tabla para conducirlo. De esta forma, realizará un barrido secuencial sobre la tabla no indexada y, por cada fila, accederá por índice a la otra. La situación inversa obligaría a una lectura secuencial de toda la tabla no indexada por cada fila leida en la tabla indexada.

Por ejemplo, si la columna X está indexada en la tabla TAB_1 pero no lo está en la tabla TAB_2, en ambas consultas:

```
SELECT T1.X,                    SELECT T1.X,
T2.Y                            T2.Y,
FROM TAB_1 T1,                  FROM TAB_1 T1,
TAB_2 T2                        TAB_2 T2
WHERE T1.X = T2.X               WHERE T2.X = T1.X
```

la tabla conductora será TAB_2. Una excepción a esta regla se da cuando uno de los predicados sobre la tabla no indexada garantiza que se va a recuperar una sola fila (por ejemplo, buscando por ROWID) (Elmasri & Navathe, 2010).

2.1.9 Plan de Ejecución

Cada vez que se ejecuta una consulta en un motor de bases de datos, internamente se ejecutan una serie de operaciones, que varían según la consulta, los datos y obviamente, el motor de base de datos. El conjunto de pasos que tiene que realizar el motor para ejecutar la consulta, se llama Plan de Ejecución (Grimpi IT Blog).

El plan de ejecución lo genera un componente del motor de la base de datos llamado Optimizador de consultas durante la fase de optimización del

procesamiento de la consulta, esto tiene en cuenta muchos factores diferentes, como los predicados de búsqueda en la consulta, las tablas implicadas y sus condiciones de combinación, la lista de columnas devueltas y la presencia de índices útiles que pueden servir como rutas de acceso eficaces a los datos (Pilecki, 2007).

2.1.9.1 Operaciones utilizadas en el plan de ejecución de PostgreSQL 9.2

2.1.9.1.1 Index and Table Access

2.1.9.1.1.1 Seq Scan.

La operación Seq Scan explora toda la relación (tabla) como se almacena en el disco (como TABLE ACCESS FULL).

2.1.9.1.1.2 Index Scan.

El Index Scan realiza un recorrido B-tree, camina a través de los nodos de la hoja de encontrar todas las entradas coincidentes, y obtiene los datos de las tablas correspondientes. Es como un rango de índice seguido de un cuadro de acceso por rowid operación index.

2.1.9.1.1.3 Index Only Scan.

Index Only Scan realiza un recorrido B-tree y camina a través de los nodos de la hoja para encontrar todas las entradas que coincidan. No hay acceso a la tabla necesaria debido a que el índice tiene todas las columnas para satisfacer la consulta.

2.1.9.1.2 Join Operations.

En general, las operaciones de combinación procesan sólo dos tablas a la vez. En el caso de una consulta tiene más combinaciones, son ejecutados de forma

secuencial: primero dos tablas, el resultado intermedio con la siguiente tabla. En el contexto de las uniones, el término "tabla" podría por lo tanto, también significa "resultados intermedios".

2.1.9.1.2.1 Nested Loops.

Combina dos tablas por ir a buscar el resultado de una tabla y la consulta de la otra tabla para cada fila de la primera.

2.1.9.1.2.2 Hash Join / Hash.

Las Hash Join cargan los registros de candidatos de un lado de la unión en una tabla hash (marcado con hash en el plan) que luego se probaron para cada registro desde el otro lado de la combinación.

2.1.9.1.2.3 Merge Join.

El (sort) Merge Join combina dos listas ordenadas. Ambos lados de la unión debe ser clasificado previamente.

2.1.9.1.3 Sorting and Grouping

2.1.9.1.3.1 Sort / Sort Key.

Ordena el conjunto de las columnas mencionadas en Sort Key. La operación de Ordenar necesita grandes cantidades de memoria para materializar el resultado intermedio (no segmentado).

2.1.9.1.3.2 GroupAggregate.

Áridos un preclasificado establecen de acuerdo con la cláusula group by. Esta operación no almacena grandes cantidades de datos.

2.1.9.1.3.3 HashAggregate.

Utiliza una tabla hash temporal para agrupar los registros. La operación HashAggregate no requiere un conjunto de datos clasificados previamente, sino que utiliza grandes cantidades de memoria para materializar el resultado intermedio (no segmentado). La salida no se ordena de una manera significativa (Winand, 2011).

2.1.9.2 Operaciones utilizadas en el plan de ejecución de SQL Server Developer 2008 R2

2.1.9.2.1 Index and Table Access.

2.1.9.2.1.1 Table Scan.

Significa que el motor tiene que leer toda la tabla. Esto solo puede suceder cuando la tabla es Heap (no tiene un índice clustered). En algunos casos, cuando es una tabla pequeña un Table Scan es la mejor opción, ya que produce poco overhead. De hecho la tabla puede tener índices y sin embargo el SQL elige usar un table scan porque sería más rápido. Pero cuando la tabla es más grande, no debería haber Table Scan, ya que es muy costoso. Para solucionar este problema, hay ver si la tabla tiene índices y si se están usando correctamente. Lo importante es prestarle atención cuando vemos un table Scan. Muchas veces, nuestros problemas de performance pasan por ahí.

2.1.9.2.1.2 Clustered Index Scan.

Esta operación es muy similar a un table scan, el motor recorre toda la tabla. La diferencia entre uno y otro, es que el Clustered Index Scan se realiza en una tabla que tiene un índice Clustered y el Table Scan en una tabla que no tiene este tipo de índice.

Otra vez tenemos que evaluar si esta opción es la que realmente queremos. Muchas veces, por un mal uso de los índices, se ejecuta esta operación, cuando en realidad queríamos otra más eficiente.

2.1.9.2.1.3 Clustered Index Seek.

Esta operación significa que el motor está usando efectivamente el índice Clustered de la tabla.

2.1.9.2.1.4 Index Seek.

Es similar que el Clustered Index Seek, con la diferencia de que se usa un índice Non Clustered.

2.1.9.2.1.5 Index Scan.

Esta operación se ejecuta cuando se lee el índice completo de una tabla. Es preferible a un Table Scan, ya que obviamente leer un índice es más chico que una tabla. Esta operación puede ser síntoma de un mal uso del índice, aunque también puede ser que el motor haya seleccionado que esta es la mejor operación. Es muy común un Index Scan en un join o en un ORDER BY o GROUP BY.

2.1.9.2.2 Join Operations

2.1.9.2.2.1 Neested Loop Join.

Generalmente el más frecuente. Es también el algoritmo más simple de todo. Este operador físico es usado por el motor cuando tenemos un join entre 2 tablas y la cantidad de registros es relativamente baja. También aplica con cierto tipo de joins (cross joins por ejemplo).

2.1.9.2.2.2 Merge Join.

Usada cuando la cantidad de registros a comparar son grandes y están ordenados. Aun si no están ordenadas, el motor puede predecir que es más rápido ordenar la tabla.

2.1.9.2.2.3 Hash Join.

Este tipo de JOIN es muy especifico para grandes volúmenes de datos, usado generalmente cuando las tablas relacionadas no tienen índice en ninguna de los campos a comparar.

2.1.9.2.2.4 Hash Match.

Cuando ven este operador es porque el motor está compando contenido, puede aparecer en un JOIN, WHERE y son lugares donde no deberían estar, lo hacen por falta de ÍNDICE s principalmente. Donde si son muy útiles es cuando incluimos la cláusula DISTINCT, UNION, UNION ALL, en donde no solo se compara el valor de un campo, sino de todo un conjunto de columnas o incluso filas y columnas.

****

.1.9.2.3 Sorting and Grouping.

2.1.9.2.3.1 Sort.

Como el nombre lo indica, esta operación ordena. El Sort solo se hace cuando el campo o los campos que se desean ordenar, no están indexados. A veces esta operación se ejecuta sola, sin que nosotros hayamos puesto en la consulta el ORDER BY, porque el motor necesita ordenar los datos por alguna razón, por ejemplo, para ejecutar un Merge Join.

2.1.9.2.3.2 Stream Aggregate.

Este tipo de operaciones ocurre cuando se llama a una función de agregación, como MIN, COUNT, MAX, SUM, entre otros. El operador Stream Aggregate requiere que la información esté ordenada por las columnas dentro de sus grupos. Primero, el optimizador ordenará si los datos no están ordenados por un operador Sort anterior. En cierta manera, el Stream Aggregate es similar al Merge Join, en cuanto a en que situaciones se produce.

2.1.9.2.3.3 Hash Match (Aggregate).

Hay que tener cuidado cuando vemos este operador. Esta operación también ocurre cuando se llama a funciones de agregación del tipo MIN, COUNT, AVG, etc. Así como el Stream Aggregate es comparable al Merge Join, el Hash Match Aggregate es similar al Hash Join. Lo que hace internamente es armar una tabla de hash. En situaciones donde la cantidad de registros es elevada o no se están indexadas las columnas por las cuales agrupa la consulta, el motor del SQL va a elegir esta operación (Winand, 2011).

2.1.9.3 Operaciones utilizadas en el plan de ejecución de Oracle 11gR2.

2.1.9.3.1 Index and Table Access

2.1.9.3.1.1 Index Unique Scan.

Sólo realiza el recorrido B-tree. La base de datos utiliza esta operación si una restricción de unicidad asegura que los criterios de búsqueda coincidirán con no más de una entrada.

2.1.9.3.1.2 Index Range Scan.

La gama exploración de índice realiza el recorrido B-tree y sigue la cadena nodo hoja para encontrar todas las entradas que coincidan.

El también llamado index filter predicates a menudo causan problemas de rendimiento de una gama exploración de índice.

2.1.9.3.1.3 Index Full Scan.

Lee el índice de todas las filas para todo en el índice. Según diversas estadísticas del sistema, la base de datos puede realizar esta operación si necesita todas las filas en el índice, a causa de la correspondiente orden por cláusula. En cambio, el optimizador puede también utilizar un índice de Búsqueda rápida FULL SCAN y realizar una operación de ordenación adicional.

2.1.9.3.1.4 Index Fast Full Scan.

Lee todo el índice de todas las filas, tal como se almacena en el disco. Esta operación se realiza generalmente en lugar de un escaneo completo de tabla, si todas las columnas necesarias están disponibles en el índice. Similar a CUADRO DE ACCESO COMPLETO, el índice de Búsqueda rápida FULL SCAN puede beneficiarse de las operaciones de lectura de varios bloques.

2.1.9.3.1.5 Table Access By Index Rowid.

Recupera una fila de la tabla con el ROWID recuperado de la búsqueda por índice anterior.

2.1.9.3.1.6 Table Access Full.

Esto también es conocido como escaneo completo de tabla. Lee toda la tabla, todas las filas y columnas, como almacenados en el disco. Aunque varios bloques operaciones de lectura mejoran la velocidad de un escaneo completo de tabla considerablemente, sigue siendo una de las operaciones más caras. Además de las altas tasas IO, un escaneo completo de tabla debe inspeccionar todas las filas de la tabla por lo que también puede consumir una cantidad considerable de tiempo de CPU.

2.1.9.3.2 Join Operations.

2.1.9.3.2.1 Nested Loops Join.

Combina dos tablas por ir a buscar el resultado de una tabla y la consulta de la otra tabla para cada fila de la primera.

2.1.9.3.2.2 Hash Join.

Las cargas uniones hash los registros de candidatos de un lado de la unión en una tabla hash que a continuación se probaron para cada fila desde el otro lado de la combinación.

2.1.9.3.2.3 Merge Join.

La unión de fusión combina dos listas ordenadas. Ambos lados de la unión debe ser clasificado previamente.

2.1.9.3.3 Sorting and Grouping.

2.1.9.3.3.1 Sort Order By.

Clasifica el resultado de acuerdo con el orden de la cláusula. Esta operación necesita grandes cantidades de memoria para materializar el resultado intermedio (no segmentado).

2.1.9.3.3.2 Sort Order By Stopkey.

Ordena un subconjunto de los resultados de acuerdo con la cláusula order by. Se utiliza para las consultas top-N.

2.1.9.3.3.3 Sort Group By.

Ordena el conjunto de resultados en el grupo de columnas y agrega el resultado ordenados en una segunda etapa. Esta operación necesita grandes cantidades de memoria para materializar el conjunto de resultados intermedios (no segmentado).

2.1.9.3.3.4 Sort Group By Nosort.

Áridos un preclasificado establecen de acuerdo al grupo por cláusula. Esta operación no almacena el resultado intermedio: se ejecuta de forma segmentada.

2.1.9.3.3.5 Hash Group By.

Agrupa el resultado utilizando una tabla hash. Esta operación necesita grandes cantidades de memoria para materializar el conjunto de resultados intermedios (no segmentado). La salida no se ordena de una manera significativa.

2.1.9.3.3.6 Count Stopkey.

Una operación count donde el número de filas devueltas está limitado por la expresión ROWNUM en la cláusula WHERE (Winand, 2011).

2.2 Hints

En forma predeterminada, el SGBD considera el plan de ejecución determinado por el optimizador. Sin embargo, por medio de hints se puede inducir a que el SGBD ejecute una sentencia con métodos deseados por el usuario. Los hints se colocan en la sentencia SQL a ejecutar. La sintaxis es:
SELECT /*+ [HINTS]*/ [columnas] FROM…
Si en la consulta se usan alias para las tablas, estos se deben usar en el hint (en lugar de los nombres de las tablas).

El hint que se usó en las pruebas fue: INDEX(tabla [índice]): induce al optimizador a usar el índice especificado de la tabla.

CAPITULO III

METODOLOGÍA DE LA INVESTIGACIÓN

3.1 MATERIALES

Tabla 7. Materiales

Tipo	Ítem	Características
Hardware	Laptop	Procesador Intel® Core™ i7 de tercera generación Disco Duro de 1 TB Memoria RAM de 8 GB
	Impresora	Canon Pixma 20
Software	Sistema Operativo	Windows 7 Professional
	Motores de Base de Datos	Postgresql 9.2 - pgAdmin III SQL Server Developer 2008 R2 - SQL Server Management Studio Oracle Enterprise Edition 11g R2 IDE sqldeveloper
	Utilitarios	Microsoft Office 2013
	Herramienta para la migración de datos	ESF Database Migration Toolkit - 8.0.22
	Estadístico	Rcommander

3.2 METODOLOGÍA

Se utilizó un segmento de la Base de Datos Openbravo Pos poblada con 10 y 20 millones de registros para el objetivo de la investigación; teniendo la base de datos poblada se crearon cuatro consultas, cada una de estas se escribieron optimizadas de ocho formas diferentes con el SQL Estándar.

Para la escritura de las consultas primero se utilizó la forma simple, luego se utilizó el inner join, las subquery fueron desarrolladas dentro de la cláusula

where con el operador in, dentro del from también se crearon subquery, y en el listado de los campos del select, después se realizaron combinaciones entre join y subquery. Posteriormente se aplicó índices en los campos más utilizados en la condición de búsqueda de las consultas escritas de forma simple, con join, subquery y la combinación de ambos.

Al final se tomaron los tiempos de respuestas y los planes de ejecución de cada consulta ejecutada para evaluar el rendimiento de las query en los tres motores de base de datos.

3.3 METODOLOGÍA DE COMPROBACIÓN DE HIPÓTESIS

El diseño experimental es usado en investigaciones de comportamiento, por lo que se aplicó un Diseño Completamente al Azar con arreglo factorial 2 x 3 x 8 con cuatro repeticiones. Para la comparación de medias entre los registros, DBMS y consultas se empleó la prueba de Tukey al 95% de probabilidades.

Esquema del diseño

Tabla 8. Grupo Experimental

I	R1	M1	S1	S2	S3	S4	S5	S6	S7	S8
	R1	M2	S1	S2	S3	S4	S5	S6	S7	S8
	R1	M3	S1	S2	S3	S4	S5	S6	S7	S8
II	..	..	..	..	..	..	..	..	..	..
III	..	..	..	..	..	..	..	..	..	..
IV	..	..	..	..	..	..	..	..	..	..
I	R2	M1	S1	S2	S3	S4	S5	S6	S7	S8
	R2	M2	S1	S2	S3	S4	S5	S6	S7	S8
	R2	M3	S1	S2	S3	S4	S5	S6	S7	S8
II	..	..	..	..	..	..	..	..	..	..
III	..	..	..	..	..	..	..	..	..	..
IV	..	..	..	..	..	..	..	..	..	..

Dónde:

R1: 10 millones de registros

R2: 20 millones de registros

M1: Postgres

M2: SQL Server

M3: Oracle

S1: SELECT simple

S2: SELECT usando índice

S3: SELECT usando subquery

S4: SELECT usando join

S5: SELECT usando subquery + join

S6: SELECT usando índice + join

S7: SELECT usando índice + subquery

S8: SELECT usando índice + subquery + join

3.3.1 MÉTODO DE COMPROBACIÓN DE HIPÓTESIS

Para la comprobación de la hipótesis se utilizó el análisis de varianza (ANOVA).

Tabla 9. Tabla de Anova

FUENTE DE VARIACIÓN	GL	SC	CM	Fc
Registros	1			
DBMS	2			
Consultas	7			
Registros * DBMS	2			
Registros * Consultas	7			
DBMS * Consultas	14			
Registros * DBMS * Consultas	14			
Error	144			
Total	191			

El análisis de varianza (Anova) es el procedimiento para determinar si existen diferencias significativas entre factores, niveles e interacciones (tratamientos), para lo cual se emplea el diseño experimental en función de los requermientos de la investigación.

El análisis de varianza consiste en determinar la significancia estadística de cada una de las fuentes de variación empleando para ello los valores de la tabla F(fisher) en los niveles 0,05 y 0,01 dependiendo del número de grado de libertad de cada fuente y los grados de libertad del error experimental realizando los procedimientos y cálculos estadísticos y matemáticos que demanda el análisis de varianza del diseño empleado.

Una vez realizado el análisis de varianza y haber determinado la significancia estadísticas de las fuentes se procede a realizar la comparación de medias mediante el uso de pruebas como Duncan, Tukey y otras que definen si los factores niveles y tratamientos son estadísticamente iguales o diferentes entre sí, para lo cual se emplean también tablas predefinidas y generalmente se los hace al 95% de probabilidad.

CAPITULO IV

RESULTADOS Y DISCUSIÓN

4.1 RESULTADOS

El análisis de resultado se realizó en los gestores de base datos PostgreSQL 9.2, SQL Server 2008 R2 Developer y Oracle 11g Enterprise Edition, en ellos se pudo observar el tiempo y plan de ejecución de cada consulta ejecutada.

4.1.1 INTERPRETACIÓN GENERAL DE GRÁFICOS Y TABLAS

En los gráficos mostrados a continuación, de diez millones de datos (izquierda) y 20 millones de datos (derecha), el eje de las (X) representan los gestores de base de datos, mientras que el eje de las (Y) representan los tiempos de respuestas generados por las consultas ejecutadas.

Las tablas que están a continuación de cada gráfico representan los tiempos de respuestas expuestos para los diez millones (color anaranjado) y 20 millones de datos (color verde oliva).

4.1.2 ANÁLISIS DE RESULTADOS DE SELECT SIMPLE

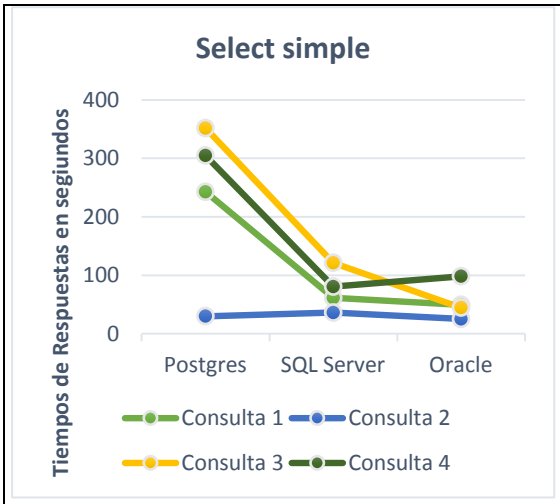


Gráfico 1. Pruebas SELECT simple con 10 millones de datos.

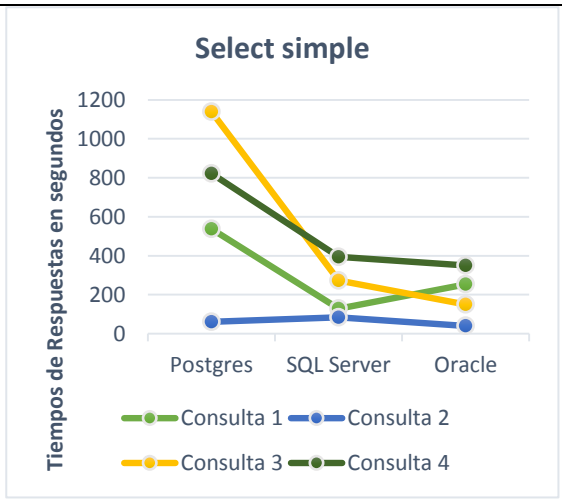


Gráfico 2. Pruebas SELECT simple con 20 millones de datos.

Tabla 10. Tiempos de Respuestas Select Simple

Postgres		SQL Server		Oracle	
10 millones	20 millones	10 millones	20 millones	10 millones	20 millones
242,57	537,86	61,2	129	49,23	254,1
29,92	61,05	36	84	25,07	40,52
351,54	1139,22	121,2	273	44,62	150,39
304,75	822,23	80,4	394,2	98,22	350,22

El gráfico 1 muestra los tiempos de respuesta de las cuatro consultas usando SELECT simple con diez millones de datos. Las consultas de selección simples se crearon con la estructura SELECT – FROM – WHERE, seleccionando los atributos en el SELECT, listando las tablas en el FROM, emparejando filas de tablas y filtrando campos de acuerdo a un valor o a un rango en la cláusula WHERE.

La consulta 1 con diez millones de datos en Oracle dio un tiempo de respuesta de 49,23 segundos seguido de SQL Server con 61,2 segundos y Postgres 242,57 segundos, con veinte millones de datos (Gráfico 2) el menor tiempo fue en SQL Server con 129 segundos, luego Oracle con un tiempo de 254,1 segundos y Postgres 537,86 segundos.

La consulta 2 con diez millones de datos en Oracle el tiempo de respuesta fue de 25,07 segundos, seguido de Postgres con 29,92 segundos y SQL Server 36 segundos. Con veinte millones de datos el menor tiempo de respuesta lo dio Oracle con 40,52 segundos seguido de Postgres 61,05 segundos y SQL Server 84 segundos.

La consulta 3 con diez millones de datos Oracle dio un tiempo de 44,62 segundos, seguido de SQL Server 121,2 segundos y Postgres 351,54 segundos. Mientras que con veinte millones los tiempos de respuestas fueron de 150,39 segundos en Oracle, 273 segundos en SQL Server y 1139 segundos en Postgres.

Se observa que en la consulta 4 con diez millones de datos el tiempo de respuesta en SQL Server fue de 80,4 segundos, en Oracle 98,22 segundos y

en Postgres 304,75 segundos. Con veinte millones de datos SQL Oracle dio un tiempo de 350,22 segundos, SQL Server 394,2 segundos y postgres 822,23 segundos.

4.1.3 ANÁLISIS DE RESULTADOS DE SELECT USANDO ÍNDICES

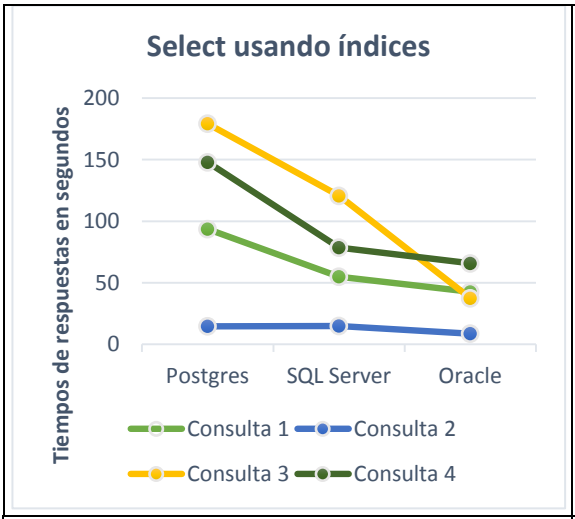


Gráfico 3. Pruebas SELECT usando índices con 10 millones de datos.

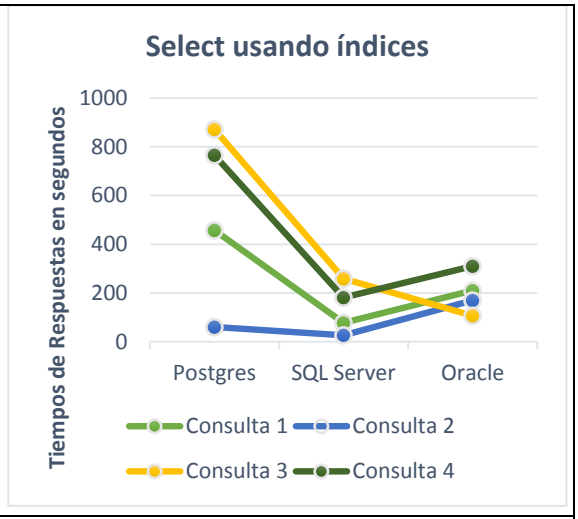


Gráfico 4. Pruebas SELECT usando índices 20 millones de datos.

Tabla 11. Tiempos de Respuestas Select usando índices

Postgres		SQL Server		Oracle	
10 millones	20 millones	10 millones	20 millones	10 millones	20 millones
93,62	457,25	55	78	42,62	210,05
14,76	60,32	15	27	8,79	169,58
179,24	871,11	120,6	258,6	37,47	106,97
147,84	765,29	78,6	181,4	65,76	310,21

El gráfico 3 muestra los tiempos de respuestas de las cuatro consultas SELECT usando índices con diez millones de datos. La consulta 1 se creó dos índices para los atributos fecha y monto ambos son usados en la cláusula WHERE el índice fue muy útil porque la cantidad de filas devueltas son mínimas en la base de datos de gran tamaño el acceso por índice es mucho más rápido que un recorrido completo de tabla. Se observa que con diez millones de datos Oracle dio un tiempo de respuesta de 42,62 segundos, SQL Server 55 segundos y Postgres 93,62 segundos. En el gráfico 4 con veinte

millones existe un cambio el menor tiempo lo dio SQL Server con 78 segundos luego Oracle con un tiempo de 210,05 segundos y Postgres 457,25 segundos.

La consulta 2 se creó un índice en un atributo foreign key, luego al ser emparejado con el primary key es mucho más eficiente buscar en los índices para localizar los registros que cumplen el criterio que tener que buscar en ambas tablas y localizar los registros mejorando así el tiempo de respuesta.

En la representación gráfica se observa que con diez millones de datos Oracle dio un tiempo de respuesta de 8,79 segundos, Postgres 14,76 segundos y SQL Server 15 segundos. Con veinte millones de datos SQL Server dio un tiempo de respuesta de 27 segundos, Postgres 60,32 segundos y Oracle 169,58 segundos.

La consulta 3 se creó un índice para el atributo fecha utilizado en la cláusula WHERE, filtrar por medio de un índice es correcto y más rápido. Los tiempos de respuestas en Oracle fueron de 37,47 segundos, SQL Server 120,6 segundos y Postgres 179,24 segundos. Para los veinte millones de datos Oracle dio un tiempo de respuesta de 106,97 segundos, SQL Server 258,6 segundos y Postgres 871,11 segundos.

La consulta 4 se creó un índice en el atributo pago usado en la cláusula WHERE para filtrar los datos de acuerdo a un rango establecido. Para los diez millones de datos Oracle dio un tiempo de respuesta de 65,79 segundos, SQL Server 78,6 segundos, Postgres 147,84 segundos. Los tiempos de respuestas con 20 millones de datos fueron en SQL Server de 181,4 segundos, Oracle 310,21 segundos y Postgres 765,29 segundos.

4.1.4 ANÁLISIS DE RESULTADOS DE SELECT USANDO SUBQUERY

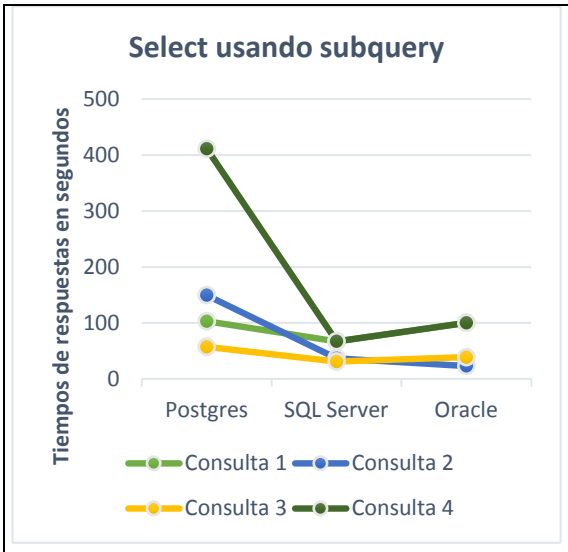


Gráfico 5. Pruebas SELECT usando subquery con 10 millones de datos.

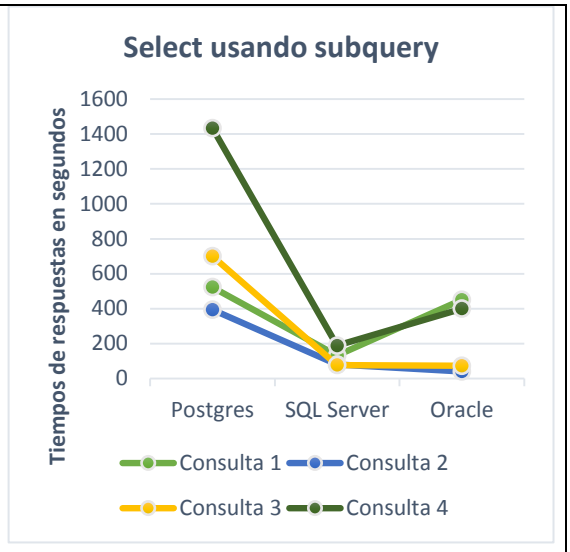


Gráfico 6. Pruebas SELECT usando subquery con 20 millones de datos.

Tabla 12. Tiempos de respuestas Select usando Subquery

Postgres		SQL Server		Oracle	
10 millones	20 millones	10 millones	20 millones	10 millones	20 millones
103	523,46	67,2	132	100,52	450,91
149,63	394,15	37	80,4	23,28	40,59
57,37	700,25	31	78	38,79	73,08
411,11	1432,51	67,2	188,4	100,13	400,51

El gráfico 5 muestra el resultado de las cuatro consultas SELECT usando subquery con 10 millones de datos. La consulta 1 se utilizó subquerys en la lista de campos de la sentencia SELECT, también se creó subquerys anidadas dentro de la cláusula WHERE utilizando el operador in. Los tiempos de respuestas para diez millones de datos fueron en SQL Server 67,2 segundos Oracle 100,52 segundos y Postgres 103 segundos. Con veinte millones de datos (gráfico 6) en SQL Server el tiempo de respuesta fue de 132 segundos, Oracle 450,91 segundos, y en Postgres 523,46 segundos.

La consulta 2 se creó una subquery dentro de la cláusula FROM que retornan un conjunto de registros de varios campos en lugar de una tabla, para diez millones de datos el tiempo de respuesta en Oracle fue de 23,28 segundos,

SQL Server 37 segundos y Postgres 149,63 segundos, para 20 millones de datos Oracle dio un tiempo de 40,59 segundos, SQL Server 80,4 segundos seguido de y Postgres 394,15 segundos.

La consulta 3 se especificó campos necesarios y una subquery en la cláusula SELECT porque al usar un group by esta técnica resulta más óptima, esto permite extraer columna que nos interese sin tener que incluirla en el GROUP BY. Los tiempos de respuestas para diez millones de datos fueron en SQL Server 31 segundos, Oracle 38,79 segundos y Postgres 57,37 segundos, para veinte millones de datos en Oracle el tiempo de respuesta fue de 73,08 segundos, SQL Server 78 segundos y en Postgres 700,25 segundos.

La consulta 4 se creó un subquery en la cláusula FROM y dentro de esta subquery se utilizaron varias subconsultas en la lista de campos de la sentencia SELECT, para diez millones de datos los tiempos de respuestas en SQL Server 67,2 segundos, Oracle 100,13 segundos, y Postgres 411,11 segundos, para 20 millones de datos el tiempo de respuesta en SQL Server 188,4 seguido de Oracle con 400,51 segundos y Postgres 1432,51 segundos.

4.1.5 ANÁLISIS DE RESULTADOS DE SELECT USANDO JOIN

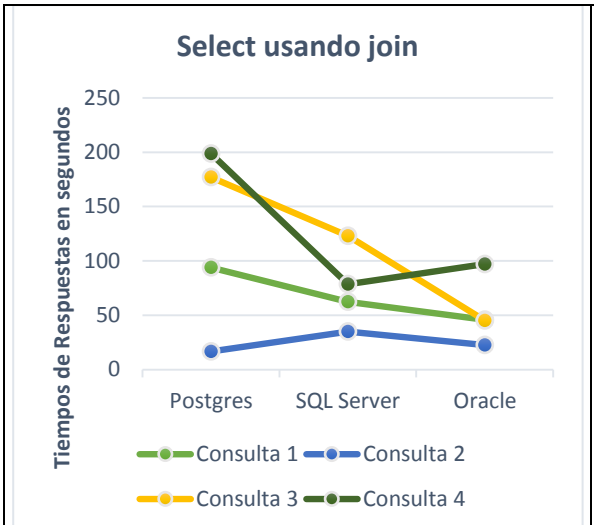


Gráfico 7. Pruebas SELECT usando join con 10 millones de datos.

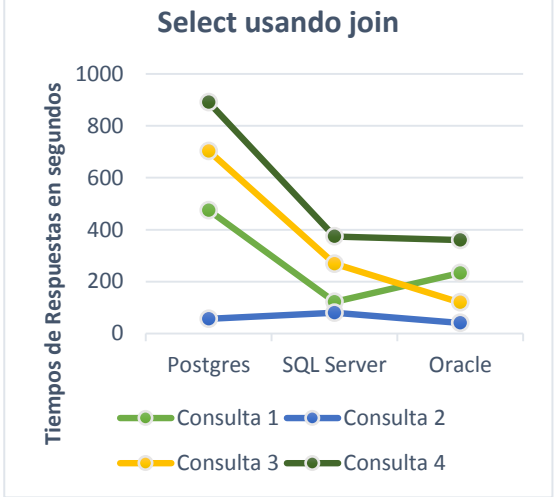


Gráfico 8. Pruebas SELECT usando join con 20 millones de datos.

Tabla 13. Tiempos de Respuestas Select usando join

Postgres		SQL Server		Oracle	
10 millones	20 millones	10 millones	20 millones	10 millones	20 millones
94,02	474,7	62,4	122,4	45,88	233,14
16,68	56,85	35	80,4	22,62	41,44
177,06	702,54	123	268,8	45,17	120,47
198,65	890,59	78,6	374,4	97,02	360,2

En el gráfico 7 muestra los tiempos de respuestas de las Select usando join con 10 millones de datos. Las cuatro consultas se escribieron haciendo el emparejamiento de tablas en la cláusula FROM utilizando inner join se utilizó la cláusula on para especificar la condición de concatenación de las columnas de diferentes tablas, se definieron varias condiciones de emparejamiento unidas por el operador and. En la consulta 1 el menor tiempo de respuesta lo dio Oracle con 45,88 segundos seguido de SQL Server 62,4 segundos y Postgres 94,02 segundos, mientras que con veinte millones (grafico 8) el menor tiempo fue en SQL Server 122,4 segundo, luego Oracle 233,14 segundos y Postgres 474,7 segundos.

La consulta 2 con diez millones de registros los tiempos de respuestas fueron en Oracle 22,62 segundos, SQL Server 35 segundos y Postgres 16,68 segundos. Con veinte millones en Oracle fue de 41,44 segundos, SQL Server 80,4 segundos y Postgres 56,85 segundos.

La consulta 3 los tiempos de respuestas con diez millones de datos en Oracle fue de 45,17 segundos, SQL Server 123 segundos y en Postgres 177,06 segundos, mientras que con 20 millones Oracle dio un tiempo de 120,47 segundos, SQL Server 268,8 segundos y Postgres 702,54 segundos.

La consulta 4 con diez millones de datos los tiempos de respuestas fueron en SQL Server 78,6 segundos, Oracle 97,02 segundos y Postgres 198,65 segundos. Con veinte millones de datos Oracle dio el menor tiempo de 360,2 segundos seguido de SQL Server 374,4 segundos y Postgres 890,59 segundos.

4.1.6 ANÁLISIS DE RESULTADOS DE SELECT USANDO SUBQUERY + JOIN

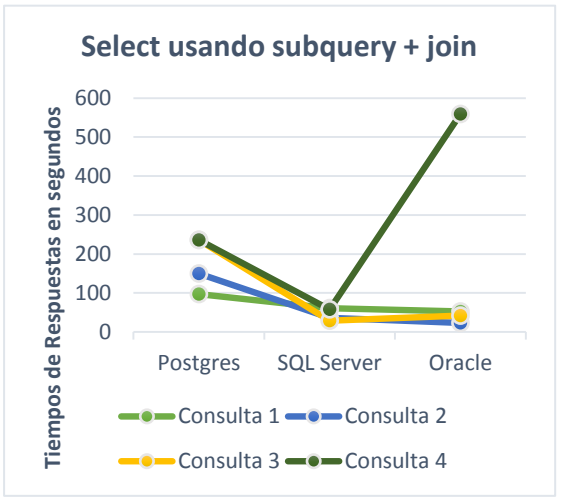


Gráfico 9. Pruebas SELECT usando subquery+ join con 10 millones de datos.

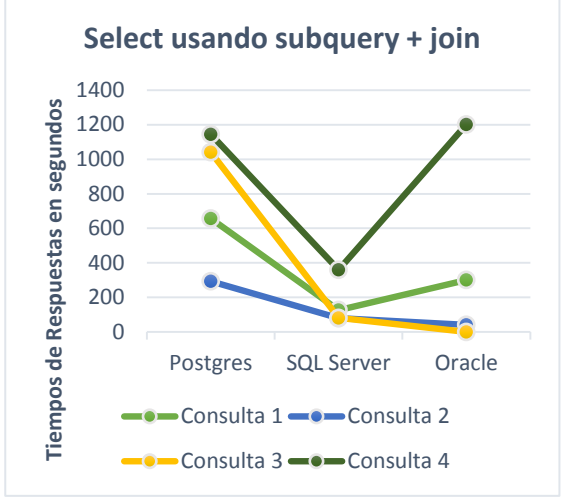


Gráfico 10.Pruebas SELECT usando subquery + join con 20 millones de datos.

Tabla 14. Tiempos de Respuestas Select usando subquery + join

Postgres		SQL Server		Oracle	
10 millones	20 millones	10 millones	20 millones	10 millones	20 millones
97,2	656,9	61,2	127,2	52,45	301,28
149,91	293,55	36	79,8	23,27	41,39
236,26	1040,72	29	81	41,91	102.27
236,11	1144,11	58	360	559,28	1201,02

En el gráfico 9 se observa los tiempos de respuestas de las cuatro consultas usando subquery + join con diez millones de datos la consulta 1 se diseñó utilizando subquerys en el lista de campos de la cláusula SELECT el emparejamiento de los registros de tablas se realizó mediante la cláusula FROM usando INNER JOIN, esta combinación de join + subquery tuvo un tiempo de respuesta de 52,45 segundos en Oracle, 61,2 segundos en SQL Server y 97,2 segundos en Postgres, con veinte millones de registros (gráfico 10) se observa que Oracle dio un tiempo de respuesta de 301,28 segundos, SQL Server 127,2 y Postgres 656,9 segundos.

La consulta 2 se diseñó creando la subquery empaquetada en la cláusula FROM, en la query interna se crearon subquerys en la lista de campos del Select, el emparejamiento de los registros de tablas se realizó mediante la cláusula FROM usando INNER JOIN se utilizó la cláusula on para especificar la condición de concatenación de las columnas de diferentes tablas. Esta combinación de subquery + join para diez millones de datos dio un tiempo de respuesta en Oracle 23,27 segundos, SQL Server 36 segundos y Postgres 149,91 segundos, con 20 millones de datos Oracle dio un tiempo de respuesta de 41,39 segundos, SQL Server 79,8 segundos y Postgres 293,55 segundos.

La consulta 3 se realizó creando la subquery en la lista de campos de la sentencia SELECT el emparejamiento de los registros de tablas se realizó mediante la cláusula FROM usando INNER JOIN, se agrupó los datos con group by, para la condición de búsqueda del grupo se utilizó having, esta combinación de join + subquery con diez millones de datos dio un tiempo de respuesta en SQL Server 29 segundos, Oracle de 41,39 segundos, Postgres 236 segundos, con 20 millones de datos el tiempo de respuesta en SQL Server fue 81 segundos, Oracle 102,27 segundos, y Postgres 1040 segundos.

La consulta 4 al igual que la consulta 3 se creó la subquery en la lista de campos de la sentencia SELECT el emparejamiento de los registros de tablas se realizó mediante la cláusula FROM usando INNER JOIN, con 10 millones de datos el tiempo de respuesta en SQL Server fue 58 segundos, Postgres 236,11 segundos y Oracle 559,28 segundos. Con 20 millones de registros los tiempos de respuesta fueron en SQL Server 360 segundos, Oracle 1201,02 segundos y Postgres 1144,11 segundos.

4.1.7 ANÁLISIS DE RESULTADOS DE SELECT USANDO ÍNDICE + JOIN

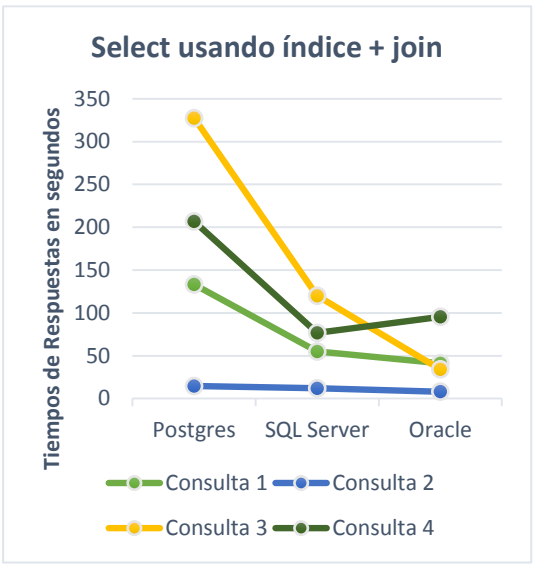


Gráfico 11. Pruebas SELECT usando índice + join con 10 millones de datos.

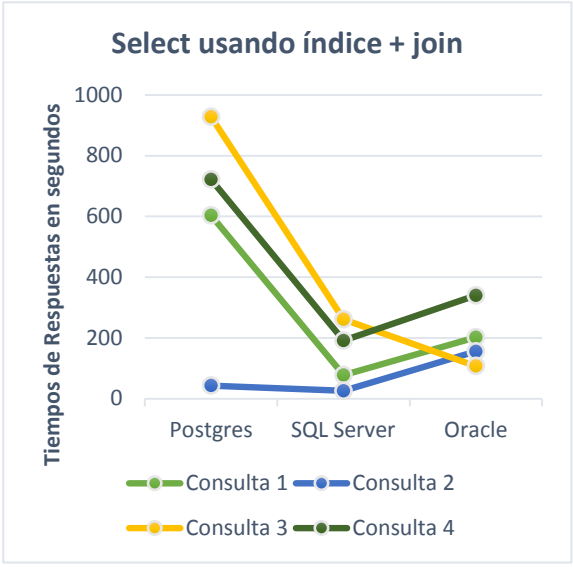


Gráfico 12. Pruebas SELECT usando índice + join con 20 millones de datos.

Tabla 15. Tiempos de Respuestas Select usando índice + join

Postgres		SQL Server		Oracle	
10 millones	20 millones	10 millones	20 millones	10 millones	20 millones
133,29	603,54	55	78	41,26	202,28
14,65	43,33	12	26	8,18	155,73
327,38	927,79	120	261,6	34,21	107,88
206,85	721,65	76,8	191,4	95,49	340,01

En el gráfico 11 muestra los tiempos de respuestas de las consultas usando índice + join con 10 millones de datos, se crearon joins con columnas indexadas. En la consulta 1 Oracle dio un tiempo de respuesta de 41,26 segundos, SQL Server 55 segundos y Postgres 133,29 segundos. Y en el gráfico 4 con veinte millones los tiempos de respuestas cambian SQL Server 78 segundos, Oracle 202,28 segundos y Postgres 603,54 segundos.

La consulta 2 en la representación gráfica se observa que con diez millones de datos Oracle dio un tiempo de respuesta de 8,18 segundos, SQL Server 12 segundos y Postgres 14,65 segundos. Con veinte millones de datos el tiempo

de respuesta en SQL Server fue 26 segundos, Postgres 43,33 segundos y Oracle 155,73 segundos.

La consulta 3 los tiempos de respuestas en Oracle fueron de 34,21 segundos SQL Server 120 segundos y Postgres 327,38 segundos. Para los veinte millones de datos Oracle dio un tiempo de respuesta de 107,88 segundos, SQL Server 261,6 segundos y 927,79 segundos.

La consulta 4 con diez millones de datos los tiempos de respuestas fueron en SQL Server 76,8 segundos, Oracle 95,49 segundos y Postgres 206,85 segundos. Los tiempos de respuestas con los 20 millones de datos en SQL Server 191,4 segundos, Oracle 340,01 segundos y Postgres 721,65 segundos.

4.1.8 ANÁLISIS DE RESULTADOS DE SELECT USANDO ÍNDICE + SUBQUERY

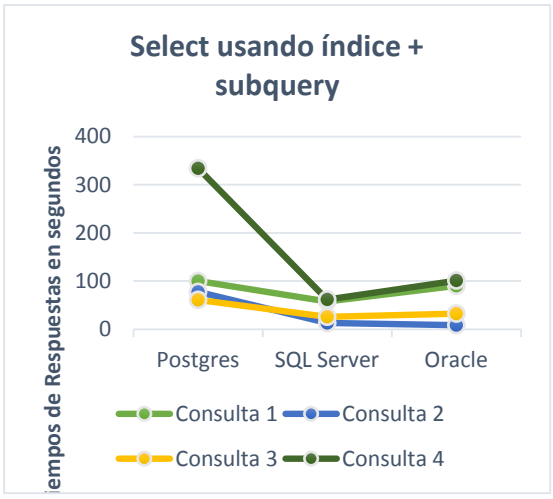


Gráfico 13. Pruebas SELECT usando índice + subquery con 10 millones de datos.

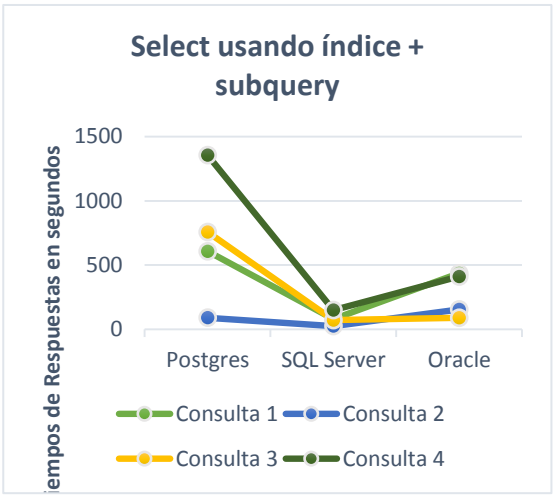


Gráfico 14. Pruebas SELECT usando índice + subquery con 20 millones de datos.

Tabla 16. Tiempos de Respuestas Select usando índice + subquery

Postgres		SQL Server		Oracle	
10 millones	20 millones	10 millones	20 millones	10 millones	20 millones
100,05	606,48	58	80,4	90,35	435,06
77,55	90,41	13	25	8,63	154,24
61,07	756,62	26	71	32,56	89,43
334,02	1355,12	61,8	149,4	101,14	410,2

El gráfico 13 con diez millones de datos muestra los tiempos de respuestas de las cuatro subquery a las cuales se le aplicó índices en los campos de búsqueda formando así la combinación índice + subquery.

La consulta 1 en SQL Server se obtuvo un tiempo de respuesta de 58 segundos, Oracle 90,35 segundos, y Postgres 100,05 segundos, mientras que con 20 millones (gráfico 14 abajo) los tiempos de respuestas fueron en SQL Server 80,4 segundos, Oracle 435,06 segundos y Postgres 606,48 segundos.

La consulta 2 con diez millones de datos se obtiene un tiempo de 8,63 segundos en Oracle, 13 segundos en SQL Server y 77,55 segundos en Postgres. Con veinte millones de datos SQL Server dio un tiempo de respuesta de 25 segundos, Postgres 90,41 segundos y Oracle 154,24 segundos.

La consulta 3 con diez millones de datos en SQL Server el tiempo de respuesta fue 26 segundos seguido de Oracle con un tiempo de 32,56 segundos y Postgres 61,07 segundos. Con veinte millones SQL Server dio un tiempo de respuesta de 71 segundos, Oracle 89,43 segundos y Postgres 756,62.

La consulta 4 con diez millones de datos en SQL Server el tiempo de respuesta fue 61 segundos, Oracle 101,14 segundos y Postgres 334,02 segundos. Con veinte millones los tiempos fueron 149,4 segundos en SQL Server, 410,2 segundos en Oracle, y 1135,12 en Postgres.

4.1.9 ANÁLISIS DE RESULTADOS DE SELECT USANDO ÍNDICE + SUBQUERY + JOIN

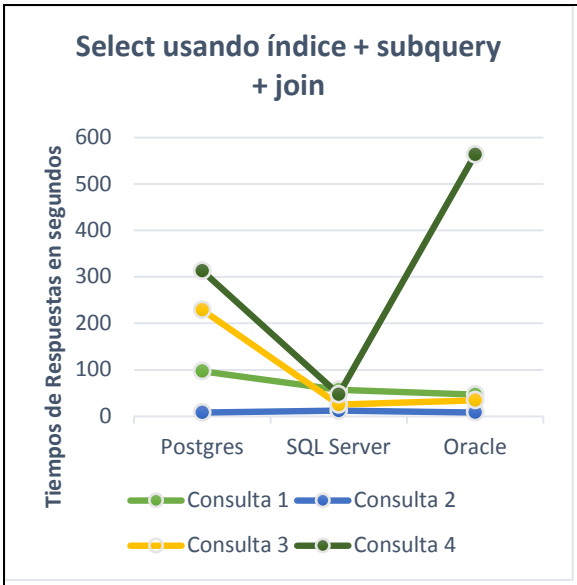


Gráfico 15. Pruebas SELECT usando índice + subquery + join con 10 millones de datos.

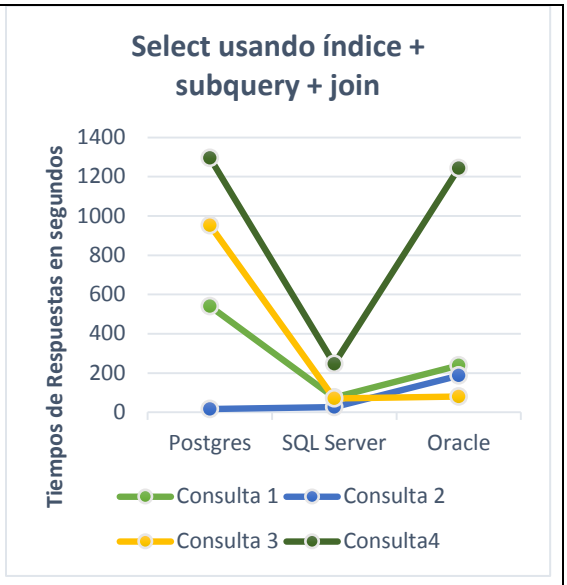


Gráfico 16. Pruebas SELECT usando índice + subquery + join con 20 millones de datos.

Tabla 17. Tiempos de Respuestas Select usando índice + subquery + join

Postgres		SQL Server		Oracle	
10 millones	20 millones	10 millones	20 millones	10 millones	20 millones
97,12	539,89	57	75	46,61	238,26
8,02	15,94	12	26	8,32	187,01
228,95	952,17	25	70,2	34,54	79,45
312,94	1295,22	47	246	563,22	1243,14

A las consultas realizadas con la combinación subquery + join se le aplicó índices en los campos de búsqueda formando de esta manera la combinación índice + subquery + join. En el gráfico 15 se observa los tiempos de respuestas de las cuatro consultas con diez millones de datos. La consulta 1 tuvo un tiempo de respuesta de 46,61 segundos en Oracle, 57 segundos en SQL Server y 97,12 segundos en Postgres, con veinte millones de registros (gráfico 16) se observa que SQL Server dio un tiempo de respuesta de 75 segundos, Oracle 238,26 segundos y Postgres 539,89 segundos.

La consulta 2 se observa que con diez millones de datos Oracle dio un tiempo de respuesta de 8,32 segundos, Postgres 8,02 segundos y SQL Server 12

segundos; con veinte millones de datos se observa que el tiempo de respuesta en Postgres fue de 15,94 segundos, SQL Server 26 segundos y Postgres 187,01 segundos.

La consulta 3 con diez millones de datos SQL Server dio un tiempo de respuesta de 25 segundos, Oracle 34,54 segundos y Postgres 228,95 segundos, los tiempos de respuesta con veinte millones de datos fueron de 70,2 segundos en SQL Server, 79,45 segundos en Oracle y 952,17 segundos en Postgres.

La consulta 4 dio un tiempo de respuesta de 47 segundos en SQL Server 312,94 segundos en Postgres y 563,22 segundos en Oracle; con veinte millones de datos los tiempos de respuesta fueron de 246 segundos en SQL Server, 1243,14 segundos en Oracle y 1295,22 segundos en Postgres.

4.2 RESULTADOS DE COMPROBACIÓN DE HIPÓTESIS Hi1

Tabla 18. Análisis de Varianza

F.V.	gl	SC	CM	F	p-valor
Registros	1	2899,76	2899,76	71,5	<0,0001
DBMS	2	2545,5	1272,75	31,38	<0,0001
Consultas	7	138,2	19,74	0,49	0,843
Registros * DBMS	2	499,73	249,86	6,16	0,0027
Registros * Consultas	7	14,39	2,06	0,05	0,9998
DBMS * Consultas	14	298,99	21,36	0,53	0,9144
Registros * DBMS * Consultas	14	70,9	5,06	0,12	>0,9999
Error	144	5840,45	40,56		
Total	191	12307,92			

4.2.1 DIMENSIÓN EFICIENCIA

Tiempos de Respuestas

En la tabla 10 el análisis de varianza determinó alta significancia estadística

para registros y DBMS y significancia para la interacción registros * DBMS; el resto de las fuentes de variación no alcanzaron significancia estadística; siendo el coeficiente de variación 49,67 %.

Tabla 19. Análisis de medias del número de registros utilizados en las pruebas.

Registros	Medias		
20 millones de datos (R2)	356,58	a	
10 millones de datos (R1)	99,97		b

Medias con una letra común no son significativamente diferentes (p > 0,05) según la prueba de Tukey.

En la tabla 19 se registran los promedios del tiempo de respuesta con respecto al número de registros.

El registro de veinte millones (R2) presentó el mayor tiempo de respuesta con 356,58 segundos, siendo según la prueba de Tukey, estadísticamente superior al registro con diez millones (R1) que presentó el menor tiempo con 99,97 segundos.

Tabla 20. Análisis de medias de los DBMS utilizados en las pruebas.

DBMS	Medias		
Postgres (M1)	408,22	a	
Oracle(M3)	174,81		b
SQL Server (M2)	101,8		b

Medias con una letra común no son significativamente diferentes (p > 0,05) según la prueba de Tukey.

En la tabla 20 se registran los promedio del tiempo de respuesta con respecto a los DBMS utilizados.

Postgres (M1) con 408,22 segundos registró el mayor tiempo, según Tukey estadísticamente superior a los DBMS Oracle (M3) y SQL Server (M2) con 174,81 y 101,8 segundos respectivamente.

Tabla 21. Análisis de medias de las ocho formas que se escribieron las consultas optimizadas

Consultas	Medias	
Subquery + Join (S5)	292,08	a
Índice + Subquery + Join (S8)	267,04	a
Simple (S1)	236,69	a
Subquery (S3)	236,69	a
Índice + Subquery (S7)	216,15	a
Índice + Join S6	199,35	a
Join (S4)	196,75	a
Índice (S2)	181,46	a

Medias con una letra común no son significativamente diferentes (p > 0,05) según la prueba de Tukey.

En la tabla 21 se registran los promedios del tiempo de respuesta de las consultas optimizadas.

La consulta subquery + join (S5) presentó el mayor tiempo con 292,08 estadisticamente igual a las demás consultas que registran tiempos entre 181,46 y 267,04 segundos; siendo el de menor tiempo el índice (S2).

Tabla 22. Análisis de medias del número de registros utilizados en los DBMS Postgres, SQL Server, Oracle

Registros / DBMS	Medias				
20 millones de datos / Postgres (R2-M1)	654,15	a			
20 millones de datos / Oracle (R2-M3)	268,78		b		
10 millones de datos / Postgres (R1-M1)	162,29		b	c	
20 millones de datos / SQL Server (R2-M2)	146,81		b	c	d
10 millones de datos / Oracle (R1-M3)	80,83			c	d
10 millones de datos / SQL Server (R1-M2)	56,79			c	

Medias con una letra común no son significativamente diferentes (p > 0,05) según la prueba de Tukey.

En la tabla 22 se registran los promedios del tiempo de respuesta de la interacción Registros * DBMS.

La interacción registros DBMS fue de mayor tiempo con 20 millones de datos en Postgres (R2-M1) con 654,15 segundos, estadísticamente superior a las

demás interacciones que registraron promedios entre 56,79 y 268,78 segundos; registrando el menor tiempo la interacción 10 millones de datos en SQL Server (R1-M2).

Tabla 23. Análisis de medias del número de registros que se utilizaron en las distintas consultas

Registros / Consultas	Medias				
20 millones de datos / Subquery + Join (R2-S5)	452,44	a			
20 millones de datos / Índice + Subquery + Join (R2-S8)	414,02	a	b		
20 millones de datos / Subquery (R2-S3)	374,52	a	b		
20 millones de datos / Simple (R2-S1)	352,98	a	b	c	
20 millones de datos / Índice + Subquery (R2-S7)	351,95	a	b	c	d
20 millones de datos / Join (R2-S4)	310,49	a	b	c	d
20 millones de datos / Índice + Join (R2-S6)	304,93	a	b	c	d
20 millones de datos / Índice (R2-S2)	291,32	a	b	c	d
10 millones de datos / Subquery + Join (R1-S5)	131,72	a	b	c	d
10 millones de datos / Simple (R1-S1)	120,39	a	b	c	d
10 millones de datos / Índice + Subquery + Join (R1-S8)	120,06		b	c	d
10 millones de datos / Subquery (R1-S3)	98,85		b	c	d
10 millones de datos / Índice + Join (R1-S6)	93,76		b	c	d
10 millones de datos / Join (R1-S4)	83,01		b	c	d
10 millones de datos / Índice + Subquery (R1-S7)	80,35			c	d
10 millones de datos / Índice (R1-S2)	71,61				d

Medias con una letra común no son significativamente diferentes (p > 0,05) según la prueba de Tukey.

En la tabla 23 se registran los promedios del tiempo de respuesta de la interacción Registros * Consultas.

La interacción registros consultas fue de mayor tiempo con 20 millones de datos utilizando subquery + join (R2-S5) con 452,44 segundos, estadísticamente superior a las demás interacciones que registraron promedios entre 71,61 y 414,02 segundos; registrando el menor tiempo la interacción 10 millones de datos usando índice (R1-S2).

Tabla 24. Análisis de medias de los DBMS Postgres, SQL Server y Oracle en los que se ejecutaron las consultas optimizadas

DBMS / Consultas	Medias	
Postgres / Subquery + Join (M1-S5)	481,85	a
Postgres / Subquery (M1-S3)	471,44	a
Postgres / Simple (M1-S1)	436,14	a b
Postgres / Indice + Subquery + Join (M1-S8)	431,28	a b
Postgres / Índice + Subquery (M1-S7)	422,67	a b
Postgres / Índice + Join (M1-S6)	372,31	a b
Postgres / Join (M1-S4)	326,39	a b
Postgres / Índice (M1-S2)	323,68	a b
Oracle / Índice + Subquery + Join (M3-S8)	300,07	a b
Oracle / Subquery + Join (M3-S5)	290,36	a b
Oracle / Índice + Subquery (M3-S7)	165,2	a b
Oracle / Subquery (M3-S3)	153,48	a b
SQL Server / Simple (M2-S1)	147,38	a b
SQL Server / Join (M2-S4)	143,13	a b
Oracle / Simple (M3-S1)	126,55	a b
Oracle / Índice + Join (M3-S6)	123,13	a b
Oracle / Join (M3-S4)	120,74	a b
Oracle / Índice (M3-S2)	118,93	a b
SQL Server / Subquery + Join (M2-S5)	104,03	a b
SQL Server / Índice + Join (M2-S6)	102,6	a b
SQL Server / Índice (M2-S2)	101,78	a b
SQL Server / Subquery (M2-S3)	85,15	a b
SQL Server / Índice + Subquery + Join (M2-S8)	69,77	b
SQL Server / Índice + Subquery (M2-S7)	60,57	b

Medias con una letra común no son significativamente diferentes (p > 0,05) según la prueba de Tukey.

En la tabla 24 se muestran los promedios del tiempo de respuesta de la interacción DBMS * Consultas.

La interacción DBMS consultas fue de mayor tiempo en Postgres usando subquery + join (M1-S5) con 481,85 segundos, estadísticamente superior a las demás interacciones que registraron promedios entre 60,57 y 471,44 segundos; registrando el menor tiempo la interacción SQL Server utilizando índice + subquery (M2-S7).

Tabla 25. Análisis de medias del número de registros utilizados en los DBMS Postgres, SQL Server, Oracle en donde se ejecutaron las consultas optimizadas

Registros / DBMS / Consultas	Medias			
20 millones de datos / Postgres / Subquery + Join (R2-M1-S5)	783,82	a		
20 millones de datos / Postgres / Subquery (R2-M1-S3)	762,59	a		
20 millones de datos / Postgres / Índice + Subquery (R2-M1-S7)	702,16	a	b	
20 millones de datos / Postgres / Índice + Subquery + Join (R2-M1-S8)	700,81	a	b	c
20 millones de datos / Postgres / Simple (R2-M1-S1)	640,09	a	b	c
20 millones de datos / Postgres / Índice + Join (R2-M1-S6)	574,08	a	b	c
20 millones de datos / Postgres / Índice (R2-M1-S2)	538,49	a	b	c
20 millones de datos / Postgres / Join (R2-M1-S4)	531,17	a	b	c
20 millones de datos / Oracle / Índice + Subquery + Join (R2-M3-S8)	436,97	a	b	c
20 millones de datos / Oracle / Subquery + Join (R2- M3-S5)	411,49	a	b	c
20 millones de datos / Oracle / Índice + Subquery (R2-M3-S7)	272,23	a	b	c
20 millones de datos / Oracle / Subquery (R2-M3-S3)	241,27	a	b	c
10 millones de datos / Postgres / Simple (R1- M1-S1)	232,2	a	b	c
20 millones de datos / SQL Server / Simple (R2-M2-S1)	220,05	a	b	c
20 millones de datos / SQL Server / Join (R2-M2-S4)	211,5	a	b	c
20 millones de datos / Oracle / Índice + Join (R2-M3-S6)	201,48	a	b	c
20 millones de datos / Oracle / Índice (R2-M3-S2)	199,2	a	b	c
20 millones de datos / Oracle / Simple (R2-M3-S1)	198,81	a	b	c
20 millones de datos / Oracle / Join (R2-M3-S4)	188,81	a	b	c
10 millones de datos / Postgres / Subquery (R1-M1-S3)	180,28	a	b	c
10 millones de datos / Postgres / Subquery + Join (R1-M1-S5)	179,87	a	b	c
10 millones de datos / Postgres / Índice + Join (R1-M1-S6)	170,54	a	b	c
10 millones de datos / Oracle / Subquery + Join (R1-M3-S5)	169,23	a	b	c
10 millones de datos / Oracle / Índice + Subquery + Join (R1-M3-S8)	163,17	a	b	c
20 millones de datos / SQL Server / Subquery + Join (R2-M2-S5)	162	a	b	c
10 millones de datos / Postgres / Índice + Subquery + Join (R1-M1-S8)	161,76	a	b	c
10 millones de datos / Postgres / Índice + Subquery (R1-M1-S7)	143,17	a	b	c
20 millones de datos / SQL Server / Índice + Join (R2-M2-S6)	139,25	a	b	c
20 millones de datos / SQL Server / Índice (R2-M2-S2)	136,25	a	b	c
10 millones de datos / Postgres / Join (R1-M1-S4)	121,6	a	b	c
20 millones de datos / SQL Server / Subquery (R2-M2-S3)	119,7	a	b	c
10 millones de datos / Postgres / Índice (R1-M1-S2)	108,87	a	b	c
20 millones de datos / SQL Server / ÍNDICE + Subquery + Join (R2-M2-S8)	104,3	a	b	c
20 millones de datos / SQL Server / Índice + Subquery (R2-M2-S7)	81,45		b	c
10 millones de datos / SQL Server / Join (R1-M2-S4)	74,75		b	c
10 millones de datos / SQL Server / Simple (R1-M2-S1)	74,7		b	c
10 millones de datos / SQL Server / Índice (R1-M2-S2)	67,3		b	c
10 millones de datos / SQL Server / Índice + Join (R1-M2-S6)	65,95		b	c
10 millones de datos / Oracle / Subquery (R1-M3-S3)	65,68		b	c
10 millones de datos / Oracle / Índice + Subquery (R1-M3-S7)	58,17		b	c

10 millones de datos / Oracle / Simple (R1-M3-S1)	54,29	b	c
10 millones de datos / Oracle / Join (R1-M3-S4)	52,67	b	c
10 millones de datos / SQL Server / Subquery (R1-M2-S3)	50,6	b	c
10 millones de datos / SQL Server / Subquery + Join (R1-M2-S5)	46,05	b	c
10 millones de datos / Oracle / Índice + Join (R1-M3-S6)	44,78		c
10 millones de datos / SQL Server / Índice + Subquery (R1-M2-S7)	39,7		c
10 millones de datos / Oracle / Índice (R1-M3-S2)	38,66		c
10 millones de datos / SQL Server / ÍNDICE + Subquery + Join (R1-M2-S8)	35,25		c

Medias con una letra común no son significativamente diferentes (p > 0,05) según la prueba de Tukey.

En la tabla 25 muestran los promedios del tiempo de respuesta de la interacción Registros * DBMS * Consultas.

La interacción registros DBMS consultas fue de mayor tiempo con veinte millones de registros en Postgres usando subquery + join (R2-M1-S5) con 783,82 segundos, estadísticamente superior a las demás interacciones que registraron promedios entre 35,25 y 762,59 segundos; registrando el menor tiempo la interacción con diez millones de datos SQL Server utilizando índice + subquery + join (R1-M2-S8).

4.2.2 DIMENSIÓN UTILIZACIÓN DE RECURSOS.

Los recursos utilizados por cada DBMS se observan a través de los planes de ejecución en dónde se muestra la forma que el DBMS accede a los datos, los métodos de unión que emplea, el número de filas estimadas, los costos estimados de las consultas en términos de operaciones de entrada-salida requeridas, requerimientos de cpu y otros factores.

Las tablas a continuación muestran el análisis de los recursos utilizados de las cuatro consultas con sus ocho diferentes formas que se escribieron en cada DBMS con 10 y 20 millones de registros.

Tabla 26. Análisis de los recursos utilizados de la consulta 1 en Postgres

Descripción	SELECT		Tablas utilizadas								Operaciones utilizadas en el Plan de Ejecución							
											Acceso a Tablas e Índices				Métodos de unión			
			atributoconjuntoinstancia	ingresos	lineasimpuesto	lineasticket	persona	productos	roles	tickets	Seq Scan	Index Scan	Bitmap Heap Scan	Bitmap Index Scan	Nested Loops Semi Join	Hash	Hash Join	Hash Semi Join
Se desea obtener un informe completo de las ventas realizadas por los cajeros en un rango de fecha y cuyo total de venta sean inferiores a los $500 dólares, de las ventas se debe mostrar lo siguiente: nombre de los productos, atributos, categoría, precio, iva; y de los cajeros se debe mostrar el monto.	Simple	Nº Filas	1106	845641	9974621	29999268	10	78763	1	10000000	8					7	7	
		Costo	0.00..21.06	0.00..213695	0.00..188704	0.00..520575	0.00..1.10	0.00..1794	0.00..1.04	0.00..154055								
		Nº Filas	1106	1707491	19940462	59996160	10	78763	1	20000000								
		Costo	0.00..21.06	0.00..427389.00	0.00..377406.00	0.00..1041110.60	0.00..1.10	0.00..1794	0.00..1.04	0.00..308109.00								
	Índice	Nº Filas	1106	846618	9974621	29999268	10	78763	1	10000000	7		1	1		7	7	
		Costo	0.00..21.06	0.00..17755.5	0.00..188704	0.00..520575	0.00..1.10	0.00..1794	0.00..1.04	0.00..154055								
		Nº Filas	1106	1709452	19940462	59996160	10	78763	1	20000000								
		Costo	0.00..21.06	0.00..35848.96	0.00..377406.00	0.00..1041110.60	0.00..1.10	0.00..1794	0.00..1.04	0.00..308109.00								
	Subquery	Nº Filas	1	845641	9974621	29999268	10	1	1	10000000	7	3			2	5		5
		Costo	0.00..8.27	0.00..213695	0.00..188704	0.00..520575	0.00..12.40	0.00..8.28	0.00..1.04	0.00..154055								
		Nº Filas	1	1707491	19940462	59996160	10	1	1	20000000								
		Costo	0.00..8.27	0.00..427389.00	0.00..377406.00	0.00..1041110.60	0.00..12.40	0.00..8.28	0.00..1.04	0.00..308109.00								
	Join	Nº Filas	1106	845641	9974621	29999268	10	78763	1	10000000	8					7	7	
		Costo	0.00..21.06	0.00..213695	0.00..188704	0.00..520575	0.00..1.10	0.00..1794	0.00..1.04	0.00..154055								
		Nº Filas	1106	1707491	19940462	59996160	10	78763	1	20000000								
		Costo	0.00..21.06	0.00..427389.00	0.00..377406.00	0.00..1041110.60	0.00..1.10	0.00..1794	0.00..1.04	0.00..308109.00								
	Subquery + Join	Nº Filas	1	845641	9974621	29999268	5	1	1	10000000	6	2				4	4	
		Costo	0.00..8.27	0.00..213695	0.00..188704	0.00..520575	0.00..6.36	0.00..8.28	0.00..1.04	0.00..154055								
		Nº Filas	1	1707491	19940462	59996160	5	1	1	20000000								
		Costo	0.00..8.27	0.00..427389.00	0.00..377406.00	0.00..1041110.60	0.00..6.36	0.00..8.28	0.00..1.04	0.00..308109.00								
	Índice + join	Nº Filas	1106	846618	9974621	29999268	10	78763	1	10000000	7		1	1		7	7	
		Costo	0.00..21.06	0.00..17755.5	0.00..188704	0.00..520575	0.00..1.10	0.00..1794	0.00..1.04	0.00..154055								
		Nº Filas	1106	1709452	19940462	59996160	10	78763	1	20000000								
		Costo	0.00..21.06	0.00..35848.96	0.00..377406.00	0.00..1041110.60	0.00..1.10	0.00..1794	0.00..1.04	0.00..308109.00								
	Índice + Subquery	Nº Filas	1	846618	9974621	29999268	10	1	1	10000000	5	3	2	2		5		5
		Costo	0.00..8.27	0.00..17755.53	0.00..188704	0.00..520575	0.00..12.40	0.00..8.28	0.00..1.04	0.00..154055								
		Nº Filas	1	1709452	19940462	59996160	10	1	1	20000000								
		Costo	0.00..8.27	0.00..35848.96	0.00..377406.00	0.00..1041110.60	0.00..12.40	0.00..8.28	0.00..1.04	0.00..308109.00								
	Índice + Subquery + Join	Nº Filas	1	846618	9974621	29999268	5	1	1	10000000	5	2	1	1		4	4	
		Costo	0.00..8.27	0.00..17755.53	0.00..188704	0.00..520575	0.00..6.36	0.00..8.28	0.00..1.04	0.00..154055								
		Nº Filas	1	1709452	19940462	59996160	5	1	1	20000000								
		Costo	0.00..8.27	0.00..35848.96	0.00..377406.00	0.00..1041110.60	0.00..6.36	0.00..8.28	0.00..1.04	0.00..308109.00								

Tabla 27. Análisis de los recursos utilizados de la consulta 1 en SQL Sever

Descripción	SELECT		Tablas utilizadas								Operaciones utilizadas en el Plan de Ejecución							
											Acceso a Tablas e Índices				Método de Unión			Ordenación
			atributoconjuntoinstancia	ingresos	lineasimpuesto	lineasticket	persona	productos	roles	tickets	Clustered Index Scan	Clustered Index Seek	Index Scan	Index Seek	Nested Loops	Merge Join	Hash Match	Sort
Se desea obtener un informe completo de las ventas realizadas por los cajeros en un rango de fecha y cuyo total de venta sean inferiores a los $500 dólares, de las ventas se debe mostrar lo siguiente: nombre de los productos, atributos, categoría, precio, iva; y de los cajeros se debe mostrar el monto.	Simple	Nº Filas	1106	813053	9986423	29999270	10	78763	1	10000000	6	1	1		1		6	
		Costo	0,000343	2,750039	2,750039	8,249838	0,000168	0,021699	0,000158	2,750039								
		Nº Filas	1106	1687233	19971430	59996160	10	78763	1	20000000								
		Costo	0,000343	5,500039	5,500039	16,498980	0,000168	0,021699	0,000158	5,500039								
	Índice	Nº Filas	1106	847651	9990688	29999270	10	78763	1	1000000	3	1	1	3	2		5	
		Costo	0,000343	0,233143	2,747478	8,249838	0,000168	0,021699	0,000158	1,100157								
		Nº Filas	1106	1667076	19980870	59996160	10	78763	1	2000000								
		Costo	0,000343	0,458485	5,494778	16,498980	0,000168	0,021699	0,000158	2,200157								
	Subquery	Nº Filas	1106	813053	9986423	29999270	10	78763	1	10000000	8	1	1		2	1	7	
		Costo	0,000343	2,750039	2,750039	8,249838	0,000168	0,021699	0,000158	2,750039								
		Nº Filas	1106	1687233	19971430	59996160	10	78763	1	20000000								
		Costo	0,000343	5,500039	5,500039	16,498980	0,000168	0,021699	0,000158	5,500039								
	Join	Nº Filas	1106	813053	9986423	29999270	10	78763	1	10000000	6	1	1		1		6	
		Costo	0,000343	2,750039	2,750039	8,249838	0,000168	0,021699	0,000158	2,750039								
		Nº Filas	1106	1687233	19971430	59996160	10	78763	1	20000000								
		Costo	0,000343	5,500039	5,500039	16,498980	0,000168	0,021699	0,000158	5,500039								
	Subquery + Join	Nº Filas	1106	813053	9986423	29999270	10	78763	1	10000000	6	1	1		1		6	
		Costo	0,000343	2,750039	2,750039	8,249838	0,000168	0,021699	0,000158	2,750039								
		Nº Filas	1106	1687233	19971430	59996160	10	78763	1	20000000								
		Costo	0,000343	5,500039	5,500039	16,498980	0,000168	0,021699	0,000158	5,500039								
	Índice + join	Nº Filas	1106	847651	9990574	29999270	10	78763	1	1000000	3	1	1	3	2		5	
		Costo	0,000343	0,233143	2,747447	8,249838	0,000168	0,021699	0,000158	1,100157								
		Nº Filas	1106	1667076	19980870	59996160	10	78763	1	2000000								
		Costo	0,000343	0,458485	5,494778	16,498980	0,000168	0,021699	0,000158	2,200157								
	Índice + Subquery	Nº Filas	1106	847651	9990574	29999270	10	78763	1	799962	3	1	1	5	3		7	1
		Costo	0,000343	0,233143	2,747447	8,249838	0,000168	0,021699	0,000158	0,880115								
		Nº Filas	1106	1667076	19980870	59996160	10	78763	1	2000000								
		Costo	0,000343	0,458485	5,494778	16,498980	0,000168	0,021699	0,000158	2,200157								
	Índice + Subquery + Join	Nº Filas	1106	847651	9990574	29999270	10	78763	1	799962	3	1	1	4	2		5	
		Costo	0,000343	0,233143	2,747447	8,249838	0,000168	0,021699	0,000158	0,880115								
		Nº Filas	1106	1667076	19980870	59996160	10	78763	1	2000000								
		Costo	0,000343	0,458485	5,494778	16,498980	0,000168	0,021699	0,000158	2,200157								

Tabla 28. Análisis de los recursos utilizados de la consulta 1 en Oracle

Descripción	SELECT		Tablas Utilizadas								Operaciones utilizadas en el Plan de Ejecución							
											Acceso a Tablas e Índices				Métodos de unión			
			atributoconjuntoinstancia	ingresos	lineasimpuesto	lineasticket	persona	productos	roles	tickets	TABLE ACCES FULL	TABLE ACCES BY INDEX ROWID	INDEX RANGE SCAN	INDEX UNIQUE SCAN	HASH JOIN	HASH JOIN SEMI	HAS JOIN RIGHT SEMI	NESTED LOOPS
Se desea obtener un informe completo de las ventas realizadas por los cajeros en un rango de fecha y cuyo total de venta sean inferiores a los $500 dólares, de las ventas se debe mostrar lo siguiente: nombre de los productos, atributos, categoría, precio, iva; y de los cajeros se debe mostrar el monto.	Simple	Nº Filas	1106	860000	9965000	29000000	3	78763	1	10000000	7	1			6			1
		Costo	5	8291	10187	35701	2	208	1	8469								
		Nº Filas	1106	1721000	2904000	59000000	3	78763	1	20000000								
		Costo	5	16618	20447	71496	2	208	1	16971								
	Índice	Nº Filas	1106	860000	9965000	29000000	3	78763	1	10000000	6	2	1	1	6			1
		Costo	5	2289	10187	35690	2	208	1	8469								
		Nº Filas	1106	1721000	2904000	59000000	3	78763	1	20000000								
		Costo	5	4574	20447	71496	2	208	1	16971								
	Subquery	Nº Filas	1	8291	9965000	29000000	10	1	1	1	7	3		3	4	1	1	
		Costo	1	8291	10187	35728	1	2	1	2								
		Nº Filas	1	1721000	2904000	59000000	10	1	1	1								
		Costo	2	16618	20447	71550	1	2	1	2								
	Join	Nº Filas	1106	860000	9965000	29000000	3	78763	1	10000000	7	1		1	6			1
		Costo	5	8291	10187	35690	2	208	1	8469								
		Nº Filas	1106	1721000	2904000	59000000	3	78763	1	20000000								
		Costo	5	16618	20447	71496	2	208	1	16971								
	Subquery + Join	Nº Filas	1	860000	9965000	29000000	3	1	1	10000000	5	3		3	4			1
		Costo	1	8291	10187	35690	2	2	1	8469								
		Nº Filas	1	1721000	2904000	59000000	3	1	1	20000000								
		Costo	2	16618	20447	71496	2	2	1	16971								
	Índice + join	Nº Filas	1106	860000	9965000	29000000	3	78763	1	10000000	6	2	1	1	6			1
		Costo	5	2289	10187	35690	2	208	1	8469								
		Nº Filas	1106	1721000	2904000	59000000	3	78763	1	20000000								
		Costo	5	4574	20447	71496	2	208	1	16971								
	Índice + Subquery	Nº Filas	1	860000	9965000	29000000	3	1	1	10000000	5	5	2	3	4	1	1	
		Costo	2	2289	10187	35690	2	2	1	8469								
		Nº Filas	1	1721000	2904000	59000000	10	1	1	1								
		Costo	2	4574	20447	71550	1	2	1	2								
	Índice + Subquery + Join	Nº Filas	1	860000	9965000	29000000	3	1	1	10000000	4	4	1	3	4			1
		Costo	2	2289	10187	35690	2	2	1	8469								
		Nº Filas	1	1721000	2904000	59000000	3	1	1	20000000								
		Costo	2	4574	20447	71496	2	2	1	16971								

Tabla 29. Análisis de los recursos utilizados de la consulta 2 en Postgres

Descripción	Consulta		Tablas utilizadas						Operaciones utilizadas en el Plan de Ejecución									
									Acceso a Tablas e Índices				Métodos de Unión					Ordenación
			atributoconjuntoinstancia	categorias	clientes	lineasticket	productos	tickets	Seq Scan	Index Scan	Bitmap Heap Scan	Bitmap Index Scan	Nested Loops	Nested Loops Semi Join	Hash	Hash Join	Merge Join	Sort
Mostrar código, nombre, dirección de los clientes que han comprado productos de la categoria farmacéuticos pertenecientes a los laboratorios: QUIFATEX PROP MERZ, NATURPHARMA, GRUNENTHAL-GENERICOS, VITAFARMA, QUIFATEX GENERICOS. Mostrar el código del ticket, nombre del producto , precio de venta y las unidades vendidas.	Simple	Nº Filas	5	1	2240818	29999268	78763	10000000	5	1			1		3	3	1	1
		Costo	0.00..27	0.00..1	0.00..101640	0.00..520575	0.00..1794	0.00..154055										
		Nº Filas	5	1	2240818	59996160	78763	20000000	6				1		4	4	1	1
		Costo	0.00..27	0.00..1	0.00..65830	0.00..1041110	0.00..1794	0.00..327389										
	Índice	Nº Filas	5	1	2240818	27124	78763	10000000	4		1	1			3	3	1	1
		Costo	0.00..27	0.00..1	0.00..65830	0.00..506	0.00..1794	0.00..154055										
		Nº Filas	5	1	2240818	54246	78763	20000000										
		Costo	0.00..27	0.00..1	0.00..65830	0.00..1008	0.00..1794	0.00..308109										
	Subquery	Nº Filas	5	1	1	149996	1	1	4	8			4	1				
		Costo	0.00..27	0.00..1	0.00..8.11	0.00..280225	0.00..8,28	0.00..8.99										
		Nº Filas	5	2	1	299981	1	1										
		Costo	0.00..27	0.00..1	0.00..8.11	0.00..560428	0.00..8,28	0.00..10.26										
	Join	Nº Filas	5	1	2240818	29999268	78763	10000000	5	1			1		3	3	1	1
		Costo	0.00..27	0.00..1	0.00..101640	0.00..520575	0.00..1794	0.00..154055										
		Nº Filas	5	1	2240818	59996160	78763	20000000	6				1		4	4		
		Costo	0.00..27	0.00..1	0.00..65830	0.00..1041110	0.00..1794	0.00..308109										
	Subquery + Join	Nº Filas	5	2	1	149996	1	1	4	7			5					
		Costo	0.00..27	0.00..1	0.00..8.11	0.00..280225	0.00..8,28	0.00..8.99										
		Nº Filas	5	2	1	299981	1	1										
		Costo	0.00..27	0.00..1	0.00..8.30	0.00..560428	0.00..8,28	0.00..10.26										
	Índice + join	Nº Filas	5	1	2240818	27124	78763	10000000	4	1	1	1	1		3	3	1	1
		Costo	0.00..27	0.00..1	0.00..65830	0.00..506	0.00..1794	0.00..154055										
		Nº Filas	5	1	2240818	54246	78763	20000000										
		Costo	0.00..27	0.00..1	0.00..65830	0.00..1008	0.00..1794	0.00..308109										
	Índice + Subquery	Nº Filas	5	2	1	27124	1	1	3	8	1	1	5					
		Costo	0.00..27	0.00..1	0.00..8.11	0.00..506	0.00..8,28	0.00..8.99										
		Nº Filas	5	2	1	54246	1											
		Costo	0.00..27	0.00..1	0.00..8.30	0.00..1008	0.00..8,28	1										
	Índice + Subquery + Join	Nº Filas	5	2	1	27124	1	0.00..10.26	3	7	1	1	5					
		Costo	0.00..27	0.00..1	0.00..8.11	0.00..506	0.00..8,28	0.00..8.99										
		Nº Filas	5	2	1	54246	1	1										
		Costo	0.00..27	0.00..1	0.00..8.30	0.00..1008	0.00..8,28	0.00..10.26										

Tabla 30. Análisis de los recursos utilizados de la consulta 2 en SQL Server

Descripción	SELECT		Tablas utilizadas						Operaciones utilizadas en el Plan de Ejecución					
									Acceso Tablas e Índices			Métodos de Unión		Agrupación
			atributoconjuntoinstancia	Categoria	clientes	lineasticket	productos	ticket	Clustered Index Scan	Clustered Index Seek	Index Seek	Nested Loops	Hash Match	Stream Aggregate
Mostrar código, nombre, dirección de los clientes que han comprado productos de la categoria farmacéuticos pertenecientes a los laboratorios: QUIFATEX PROP MERZ, NATURPHARMA, GRUNENTHAL-GENERICOS, VITAFARMA, QUIFATEX GENERICOS. Mostrar el código del ticket, nombre del producto , precio de venta y las unidades vendidas.	Simple	Nº Filas	5	1	2240818	29999270	78763	10000000	5		1	1	4	
		Costo	0,0013736	0,0001581	0,6162642	8,249838	0,02169907	2,750039						
		Nº Filas	5	1	2240818	59996160	78763	20000000						
		Costo	0,0013736	0,0001581	0,6162642	16,49898	0,02169907	5,500039						
	Índice	Nº Filas	5	1	2240818	27124	78763	10000000	4		2	2	3	
		Costo	0,0003434	0,0001581	0,6162642	0,02999352	0,02169907	2,750039						
		Nº Filas	5	1	2240818	54246	78763	20000000						
		Costo	0,0003434	0,0001581	0,6162642	0,05982769	0,02169907	5,500039						
	Subquery	Nº Filas	1106	1	2240818	29999270	78763	10000000	7	2		2		1
		Costo	0,0003434	0,0001581	0,6162642	8,249838	0,02169907	2,750039						
		Nº Filas	1106	1	2240818	59996160	78763	20000000						
		Costo	0,0003434	0,0001581	0,6162642	16,49898	0,02169907	5,500039						
	Join	Nº Filas	5	1	2240818	29999270	78763	10000000	5		1	1	4	
		Costo	0,0013736	0,0001581	0,6162642	8,249838	0,02169907	2,750039						
		Nº Filas	5	1	2240818	59996160	78763	20000000						
		Costo	0,0013736	0,0001581	0,6162642	16,49898	0,02169907	5,500039						
	Subquery + Join	Nº Filas	5	1	2240818	29999270	78763	10000000	6	2		2		1
		Costo	0,0013736	0,0001581	0,6162642	8,249838	0,02169907	2,750039						
		Nº Filas	5	1	2240818	59996160	78763	20000000						
		Costo	0,0013736	0,0001581	0,6162642	16,49898	0,02169907	5,500039						
	Índice + join	Nº Filas	5	1	2240818	27124	78763	10000000	4		2	2	3	
		Costo	0,0003434	0,0001581	0,6162642	0,02999352	0,02169907	2,750039						
		Nº Filas	5	1	2240818	54246	78763	20000000						
		Costo	0,0003434	0,0001581	0,6162642	0,05982769	0,02169907	5,500039						
	Índice + Subquery	Nº Filas	1106	1	2240818	27124	78763	10000000	6	2		3		1
		Costo	0,0003434	0,0001581	0,6162642	0,02999352	0,02169907	2,750039						
		Nº Filas	5	1	2240818	54246	78763	20000000						
		Costo	0,0003434	0,0001581	0,6162642	0,05982769	0,02169907	5,500039						
	Índice + Subquery + Join	Nº Filas	5	1	2240818	27124	78763	10000000	5	2		3	5	1
		Costo	0,0003434	0,0001581	0,6162642	0,02999352	0,02169907	2,750039						
		Nº Filas	5	1	2240818	27124	78763	20000000						
		Costo	0,0003434	0,0001581	0,6162642	0,059828	0,02169907	5,500039						

Tabla 31. Análisis de los recursos utilizados de la consulta 2 en Oracle

Descripción	SELECT		Tablas Utilizadas						Operaciones utilizadas en el Plan de Ejecución					
									Acceso a Tablas e Índices				Métodos de unión	
			atributoconjuntoinstancia	Categoria	clientes	lineasticket	productos	ticket	TABLE ACCES FULL	TABLE ACCES BY INDEX ROWID	INDEX RANGE SCAN	INDEX UNIQUE SCAN	HASH JOIN	NESTED LOOPS
Mostrar código, nombre, dirección de los clientes que han comprado productos de la categoria farmacéuticos pertenecientes a los laboratorios: QUIFATEX PROP MERZ, NATURPHARMA, GRUNENTHAL-GENERICOS, VITAFARMA, QUIFATEX GENERICOS. Mostrar el código del ticket, nombre del producto , precio de venta y las unidades vendidas.	Simple	Nº Filas	5	1	2240000	29000000	78763	10000000	5	1			4	1
		Costo	5	1	10343	35673	208	8478						
		Nº Filas	5	1	2240000	59000000	78763	20000000						
		Costo	5	1	10343	71442	208	16990						
	Índice	Nº Filas	5	1	2240000	27124	78763	10000000	4	2	1	1	3	3
		Costo	5	1	10343	24562	208	8478						
		Nº Filas	5	1	2240000	54246	78763	20000000						
		Costo	5	1	10343	49132	208	16990						
	Subquery	Nº Filas	5	2	2240000	29000000	1	10000000	4	6		6	3	1
		Costo	5	1	10343	35730	2	8478						
		Nº Filas	5	2	2240000	59000000	1	20000000						
		Costo	5	1	10343	71577	2	16990						
	Join	Nº Filas	5	1	2240000	29000000	78763	10000000	5	1		1	4	1
		Costo	5	1	10343	35673	208	8478						
		Nº Filas	5	1	2240000	59000000	78763	20000000						
		Costo	5	1	10343	71442	208	16990						
	Subquery + Join	Nº Filas	5	2	2240000	26277	1	10000000	4	5		5	3	1
		Costo	5	1	10343	24540	2	8478						
		Nº Filas	5	2	2240000	54246	1	20000000						
		Costo	5	1	10343	49132	2	16990						
	Índice + join	Nº Filas	5	1	2240000	27124	78763	10000000	4	2	1	1	3	3
		Costo	5	1	10343	24562	208	8478						
		Nº Filas	5	1	2240000	54246	78763	20000000						
		Costo	5	1	10343	49132	208	16990						
	Índice + Subquery	Nº Filas	1	2	2240000	26301	1	10000000	3	7	1	6	2	3
		Costo	2	1	10343	139	2	8478						
		Nº Filas	1	2	2240000	59000000	1	20000000						
		Costo	2	1	10343	71577	2	16990						
	Índice + Subquery + Join	Nº Filas	5	2	2240000	26277	1	10000000	3	7	1	5	2	2
		Costo	5	1	10343	24540	2	8478						
		Nº Filas	5	2	2240000	54246	1	20000000						
		Costo	5	1	10343	49132	2	16990						

Tabla 32. Análisis de los recursos utilizados de la consulta 3 en Postgres

Descripción	SELECT		Tablas utilizadas					Operaciones utilizadas en el Plan de Ejecución									
								Acceso Tablas e Índices				Métodos de Unión				Ordenación y Agrupación	
			categorias	ingresos	lineasticket	productos	tickets	Seq Scan	Index Scan	Bitmap Heap Scan	Bitmap Index Scan	Nested Loops	Hash	Hash Join	Hash Semi Join	GroupAggregate	Sort
Mostrar un listado detallando la cantidad de unidades vendidas por productos que han sido registrados durante el segundo trimestre del año 2013, indicando código, precio de venta y descripción para los productos que se vendieron entre 600 y 1000 unidades, mostrar también el código y nombre de la categoría a la cual pertenece cada producto.	Simple	Nº Filas	2	3298660	29999268	78763	10000000	5					4	4		1	1
		Costo	0.00..1	0.00..213695	0.00..520575	0.00..1794	0.00..154055										
		Nº Filas	2	6680278	59996160	78763	20000000										
		Costo	0.00..1	0.00..427389	0.00..1041110	0.00..1794	0.00..308109										
	Índice	Nº Filas	2	3298660	29999268	78763	10000000	4		1	1		4	4		1	1
		Costo	0.00..1	0.00..69171	0.00..520575	0.00..1794	0.00..154055										
		Nº Filas	2	6680278	59996160	78763	20000000										
		Costo	0.00..1	0.00..140077	0.00..1041110	0.00..1794	0.00..308109										
	Subquery	Nº Filas	2	3298660	29999268	1	10000000	4	3			1	2		2	1	1
		Costo	0.00..1	0.00..213695	0.00..520575	0.00..8.28	0.00..154055										
		Nº Filas	2	6680278	59996160	1	20000000										
		Costo	0.00..1	0.00..427389	0.00..1041110	0.00..8.28	0.00..308109										
	Join	Nº Filas	2	3298660	29999268	78763	10000000	5					4	4		1	1
		Costo	0.00..1	0.00..213695	0.00..520575	0.00..1794	0.00..154055										
		Nº Filas	2	6680278	59996160	78763	20000000										
		Costo	0.00..1	0.00..427389	0.00..1041110	0.00..1794	0.00..308109										
	Subquery + Join	Nº Filas	2	3298660	29999268	78763	10000000	5	2				3	3		1	1
		Costo	0.00..1	0.00..213695	0.00..520575	0.00..1794	0.00..154055										
		Nº Filas	2	6680278	59996160	78763	20000000										
		Costo	0.00..1	0.00..427389	0.00..1041110	0.00..1794	0.00..308109										
	Índice + join	Nº Filas	2	3298660	29999268	78763	10000000	4		1	1		4	4		1	1
		Costo	0.00..1	0.00..69171	0.00..520575	0.00..1794	0.00..154055										
		Nº Filas	2	6680278	59996160	78763	20000000										
		Costo	0.00..1	0..140077	0.00..1041110	0.00..1794	0.00..308109										
	Índice + Subquery	Nº Filas	2	3298660	29999268	1	10000000	3	3	1	1	1	2		2	1	1
		Costo	0.00..1	0.00..69171	0.00..520575	0.00..8.28	0.00..154055										
		Nº Filas	2	6680278	59996160	1	20000000										
		Costo	0.00..1	0..140077	0.00..1041110	0.00..8.28	0.00..308109										
	Índice + Subquery + Join	Nº Filas	2	3298660	29999268	78763	10000000	4	2	1	1	1	3	3		1	1
		Costo	0.00..1	0.00..69171	0.00..520575	0.00..1794	0.00..154055										
		Nº Filas	2	6680278	59996160	78763	20000000										
		Costo	0.00..1	0..140077	0.00..1041110	0.00..1794	0.00..308109										

Tabla 33.Análisis de los recursos utilizados de la consulta 3 en SQL Server

Descripción	SELECT		Tablas utilizadas					Operaciones utilizadas en el Plan de Ejecución							
								Acceso Tablas e Índices				Métodos de Unión		Ordenación y Agrupación	
			categorias	ingresos	lineasticket	productos	tickets	Clustered Index Scan	Clustered Index Seek	Index Scan	Index Seek	Nested Loops	Hash Match	Sort	Stream Aggregate
Mostrar un listado detallando la cantidad de unidades vendidas por productos que han sido registrados durante el segundo trimestre del año 2013, indicando código, precio de venta y descripción para los productos que se vendieron entre 600 y 1000 unidades, mostrar también el código y nombre de la categoría a la cual pertenece cada producto.	Simple	Nº Filas	2	3327917	29999270	78763	10000000	3		1			3	1	1
		Costo	0,0001592	2,750039	8,249838	0,02169907	2,750039								
		Nº Filas	2	6681348	59996160	78763	20000000								
		Costo	0,0001592	5,500039	16,49898	0,02169907	5,500039								
	Índice	Nº Filas	2	3317807	29999270	78763	10000000	2		1	1		3	1	1
		Costo	0,0001592	0,9124362	8,249838	0,02169907	2,750039								
		Nº Filas	2	6666306	59996160	78763	20000000								
		Costo	0,0001592	1,833273	16,49898	0,02169907	5,500039								
	Subquery	Nº Filas	1	3327917	29999270	1	10000000	2	4			4	3	1	1
		Costo	0,0001581	2,750039	8,249838	0,0001581	2,750039								
		Nº Filas	1	6681348	59996160	1	20000000								
		Costo	0,0001581	5,500039	16,49898	0,0001581	5,500039								
	Join	Nº Filas	2	3327917	29999270	78763	10000000	3		1			3		1
		Costo	0,0001592	2,750039	8,249838	0,02169907	2,750039								
		Nº Filas	2	6681348	59996160	78763	20000000								
		Costo	0,0001592	5,500039	16,49898	0,02169907	5,500039								
	Subquery + Join	Nº Filas	1	3327917	29999270	78763	10000000	2	3			3	3		1
		Costo	0,0001581	2,750039	8,249838	0,02169907	2,750039								
		Nº Filas	1	6681348	59996160	78763	20000000								
		Costo	0,0001581	5,500039	16,49898	0,02169907	5,500039								
	Índice + join	Nº Filas	2	3317807	29999270	78763	10000000	2		1	1		3	1	1
		Costo	0,0001592	0,9124362	8,249838	0,02169907	2,750039								
		Nº Filas	2	6666306	59996160	78763	20000000								
		Costo	0,0001592	1,833273	16,49898	0,02169907	5,500039								
	Índice + Subquery	Nº Filas	1	3317807	29999270	1	10000000	1	4	1	1	4	3		1
		Costo	0,0001581	0,9124362	8,249838	0,0001581	2,750039								
		Nº Filas	1	6666306	59996160	1	20000000								
		Costo	0,0001581	1,833273	16,49898	0,0001581	5,500039								
	Índice + Subquery + Join	Nº Filas	1	3317807	59996160	78763	10000000	1	3		1	3	3		3
		Costo	0,0001581	0,9124362	16,49898	0,02169907	2,750039								
		Nº Filas	1	6666306	29999270	78763	20000000								
		Costo	0,0001581	1,833273	8,249838	0,02169907	5,500039								

Tabla 34. Análisis de los recursos utilizados de la consulta 3 en Oracle

Descripción	SELECT		Tablas utilizadas					Operaciones utilizadas en el Plan de Ejecución								
								Acceso a Tablas e Índices					Métodos de Unión			Agrupación
			categorias	ingresos	lineasticket	productos	tickets	TABLE ACCES FULL	TABLE ACCES BY INDEX ROWID	INDEX RANGE SCAN	INDEX UNIQUE SCAN	INDEX FAST FULL SCAN	HASH JOIN	HAS JOIN RIGHT SEMI	NESTED LOOPS	HASH GROUP BY
Mostrar un listado detallando la cantidad de unidades vendidas por productos que han sido registrados durante el segundo trimestre del año 2013, indicando código, precio de venta y descripción para los productos que se vendieron entre 600 y 1000 unidades, mostrar también el código y nombre de la categoría a la cual pertenece cada producto.	Simple	Nº Filas	2	3439000	29000000	78763	10000000	4				1	4			1
		Costo	2	8304	35701	208	5809									
		Nº Filas	2	6878000	59000000	78763	20000000									
		Costo	2	16644	71496	208	12033									
	Índice	Nº Filas	2	3439000	29000000	78763	10000000	3	1	1		1	4			1
		Costo	2	9135	35690	208	5809									
		Nº Filas	2	6878000	59000000	78763	20000000									
		Costo	2	18267	71496	208	12033									
	Subquery	Nº Filas	2	3439000	29000000	1	10000000	2	4		4	1	1	1	1	1
		Costo	1	8304	35690	1	5809									
		Nº Filas	2	6878000	59000000	1	20000000									
		Costo	1	16644	71496	1	12033									
	Join	Nº Filas	2	3439000	29000000	78763	10000000	4				1	4			1
		Costo	2	8304	35690	208	5809									
		Nº Filas	2	6878000	59000000	78763	20000000									
		Costo	2	16644	71496	208	12169									
	Subquery + Join	Nº Filas	2	3439000	29000000	78763	10000000	3	3		3	1	3		1	1
		Costo	1	8304	35690	208	5809									
		Nº Filas	2	6878000	59000000	78763	20000000									
		Costo	1	16644	71496	208	12033									
	Índice + join	Nº Filas	2	3439000	29000000	78763	10000000	2	1	1		1	4			1
		Costo	2	9135	35690	208	5809									
		Nº Filas	2	6878000	59000000	78763	20000000									
		Costo	2	18267	71496	208	12033									
	Índice + Subquery	Nº Filas	2	3439000	29000000	1	10000000	1	5	1	4	1	1	1	1	1
		Costo	1	9135	35690	1	5809									
		Nº Filas	2	6878000	59000000	1	20000000									
		Costo	1	18267	71496	1	12033									
	Índice + Subquery + Join	Nº Filas	2	3439000	29000000	78763	10000000	2	4	1	3	1	3		1	1
		Costo	1	9135	35690	208	5809									
		Nº Filas	2	6878000	59000000	78763	20000000									
		Costo	1	18267	71496	208	12033									

Tabla 35. Análisis de los recursos utilizados de la consulta 4 en Postgres

Descripción	SELECT		Tablas Utilizadas								Operaciones utilizadas en el Plan de Ejecución						
											Acceso Tablas e Índices				Métodos de unión		
			clientes	ingresos	lineasimpuesto	lineasticket	pagos	persona	productos	tickets	Seq Scan	Index Scan	Bitmap Heap Scan	Bitmap Index Scan	Nested Loops	Hash	Hash Join
Obtener un listado de los tickets realizados durante el año 2013 indicando código del ticket, nombre del cliente, nombre del producto, cantidad, precio, impuesto, fecha y el importe total debe ser entre 500 y 800 dólares, indicar también nombre de la persona que ha realizado la venta.	Simple	Nº Filas	2240818	10000000	10000000	29999268	228725	10	78763	10000000	8					7	7
		Costo	0.00..65830	0.00..163695	0.00..163704	0.00..520575	0.00..213521	0.00..1.10	0.00..1794	0.00..154055							
		Nº Filas	2240818	20000000	20000000	59996160	437153	10	78763	20000000							
		Costo	0.00..65830	0.00..327389	0.00..327406	0.00..1041110	0.00..427043	0.00..1.10	0.00..1794	0.00..308109							
	Índice	Nº Filas	2240818	10000000	10000000	29999268	228725	10	78763	10000000	7		1	1		7	7
		Costo	0.00..65830	0.00..163695	0.00..163704	0.00..520575	0.00..4800	0.00..1.10	0.00..1794	0.00..154055							
		Nº Filas	2240818	20000000	20000000	59996160	437153	10	78763	20000000							
		Costo	0.00..65830	0.00..327389	0.00..327406	0.00..1041110	0.00..9169	0.00..1.10	0.00..1794	0.00..308109							
	Subquery	Nº Filas	1	1111111	1	20	1	1	1	1	5	5			2		
		Costo	0.00..8.51	0.00..377042	0.00..213704	0.00..43.58	0.00..188521	0.00..1.13	0.00..8.28	0.00..9.01							
		Nº Filas	1	20000000	1	59996160	437153	1	1	20000000	6	2				3	3
		Costo	0.00..8.51	0.00..327389	0.00..427406	0.00..1041110	0.00..427043	0.00..1.13	0.00..8.28	0.00..308109							
	Join	Nº Filas	2240818	10000000	10000000	29999268	228725	10	78763	10000000	8					7	7
		Costo	0.00..65830	0.00..163695	0.00..163704	0.00..520575	0.00..213521	0.00..1.10	0.00..1794	0.00..154055							
		Nº Filas	2240818	20000000	20000000	59996160	437153	10	78763	20000000							
		Costo	0.00..65830	327389	0.00..327406	0.00..1041110	0.00..427043	0.00..1.10	0.00..1794	0.00..308109							
	Subquery + Join	Nº Filas	1	10000000	10000000	29999268	228725	1	1	10000000	6	2				4	4
		Costo	0.00..8.51	0.00..163695	0.00..163704	0.00..520575	0.00..213521	0.00..1.13	0.00..8.28	0.00..154055							
		Nº Filas	1	20000000	20000000	59996160	437153	1	1	20000000							
		Costo	0.00..8.51	327389	0.00..327406	0.00..1041110	0.00..427043	0.00..1.13	0.00..8.28	0.00..308109							
	Índice + join	Nº Filas	2240818	10000000	10000000	29999268	228725	10	78763	10000000	7		1	1		7	7
		Costo	0.00..65830	0.00..163695	0.00..163704	0.00..520575	0.00..4800	0.00..1.10	0.00..1794	0.00..154055							
		Nº Filas	2240818	20000000	20000000	59996160	437153	10	78763	20000000							
		Costo	0.00..65830	327389	0.00..327406	0.00..1041110	0.00..9169	0.00..1.10	0.00..1794	0.00..308109							
	Índice + Subquery	Nº Filas	1	10000000	1	29999268	228725	1	1	10000000	5	5			2		
		Costo	0.00..8.51	0.00..163695	0.00..213704	0.00..520575	0.00..4800	0.00..1.13	0.00..8.28	0.00..154055							
		Nº Filas	1	20000000	1	59996160	437153	1	1	20000000							
		Costo	0.00..8.51	327389	0.00..427406	0.00..1041110	0.00..9169	0.00..1.13	0.00..8.28	0.00..308109							
	Índice + Subquery + Join	Nº Filas	1	10000000	10000000	29999268	228725	1	1	10000000	5	2	1	1		4	4
		Costo	0.00..8.51	0.00..163695	0.00..163704	0.00..520575	0.00..4800	0.00..1.13	0.00..8.28	0.00..154055							
		Nº Filas	1	20000000	20000000	59996160	437153	1	1	20000000							
		Costo	0.00..8.51	327389	0.00..327406	0.00..1041110	0.00..9169	0.00..1.13	0.00..8.28	0.00..308109							

Tabla 36. Análisis de los recursos utilizados de la consulta 4 en SQL Server

Descripción	SELECT		Tablas utilizadas								Operaciones utilizadas en el Plan de Ejecución								
											Acceso a Tablas e Índices					Métodos de Unión		Ordenación y Agrupación	
			clientes	ingresos	lineasimpuesto	lineasticket	pagos	persona	productos	tickets	Clustered Index Scan	Clustered Index Seek	Index Scan	Index Seek	Index Spool	Nested Loops	Hash Match	Sort	Stream Aggregate
Obtener un listado de los tickets realizados durante el año 2013 indicando código del ticket, nombre del cliente, nombre del producto, cantidad, precio, impuesto, fecha y el importe total debe ser entre 500 y 800 dólares, indicar también nombre de la persona que ha realizado la venta.	Simple	Nº Filas	1	10000000	10000000	29999270	225310,9	1	1	10000000	4	3	1			3	4		
		Costo	0,0001581	2,750039	2,750039	8,249838	2,750039	0,0001581	0,0001581	2,750039									
		Nº Filas	1	20000000	20000000	59996160	450922,4	1	1	20000000	5	3				3	4		
		Costo	0,0001581	5,500039	5,500039	16,49898	5,500039	0,0001581	0,0001581	5,500039									
	Índice	Nº Filas	1	10000000	10000000	29999270	221560,1	1	1	10000000	3	3	1	1		3	4		
		Costo	0,0001581	2,750039	2,750039	8,249838	0,06096828	0,0001581	0,0001581	2,750039									
		Nº Filas	1	20000000	20000000	59996160	442924,4	1	1	20000000	4	3		1		3	4		
		Costo	0,0001581	5,500039	5,500039	16,49898	0,1218435	0,0001581	0,0001581	5,500039									
	Subquery	Nº Filas	2240818	10000000	10000000	29999270	225310,9	10	78763	10000000	5		3		2	1	6		1
		Costo	0,6162642	2,750039	11,00016	8,249838	2,750039	0,000168	0,02169907	2,750039									
		Nº Filas	2240818	20000000	20000000	59996160	450922,4	10	78763	20000000	6		2		2	1	6		1
		Costo	0,6162642	5,500039	22,00016	16,49898	5,500039	0,000168	0,02169907	5,500039									
	Join	Nº Filas	2240818	10000000	10000000	29999270	225310,9	10	78763	10000000	5		3				7		
		Costo	0,6162642	2,750039	2,750039	8,249838	2,750039	0,000168	0,02169907	2,750039									
		Nº Filas	1	20000000	20000000	59996160	450922,4	1	1	20000000	5	3				3	4		
		Costo	0,0001581	5,500039	5,500039	16,49898	5,500039	0,0001581	0,0001581	5,500039									
	Subquery + Join	Nº Filas	2240818	10000000	10000000	29999270	225310,9	10	78763	10000000	5		3				7		
		Costo	0,6162642	2,750039	2,750039	8,249838	2,750039	0,000168	0,02169907	2,750039									
		Nº Filas	1	1	20000000	59996160	450922,4	1	1	20000000	4	4				4	3	1	
		Costo	0,0001581	0,0001581	5,500039	16,49898	5,500039	0,0001581	0,0001581	5,500039									
	Índice + join	Nº Filas	2240818	10000000	10000000	29999270	221560,1	10	78763	10000000	4		3	1			7		
		Costo	0,6162642	2,750039	2,750039	8,249838	0,06096828	0,000168	0,02169907	2,750039									
		Nº Filas	1	20000000	20000000	59996160	442924,4	1	1	20000000	4	3		1		3	4		
		Costo	0,0001581	5,500039	5,500039	16,49898	0,1218435	0,0001581	0,0001581	5,500039									
	Índice + Subquery	Nº Filas	2240818	10000000	10000000	29999270	221560,1	10	78763	10000000	4		3	1	2	1	6		1
		Costo	0,6162642	2,750039	11,00016	8,249838	0,06096828	0,000168	0,02169907	2,750039									
		Nº Filas	2240818	20000000	20000000	59996160	442924,4	10	78763	20000000	5		2	1	2	1	6		1
		Costo	0,6162642	5,500039	22,00016	16,49898	0,1218435	0,000168	0,02169907	5,500039									
	Índice + Subquery + Join	Nº Filas	2240818	10000000	10000000	29999270	221560,1	10	78763	10000000	4		3	1			7		
		Costo	0,6162642	2,750039	2,750039	8,249838	0,06096828	0,000168	0,02169907	2,750039									
		Nº Filas	1	1	20000000	59996160	442924,4	1	1	20000000	3	4		1		4	3	1	
		Costo	0,0001581	0,0001581	5,500039	16,49898	0,1218435	0,0001581	0,0001581	5,500039									

Tabla 37. Análisis de los recursos utilizados de la consulta 4 en Oracle

Descripción	SELECT		Tablas utilizadas								Operaciones utilizadas en el Plan de Ejecución				
											Acceso a Tablas e Índices				Métodos de Unión
			clientes	ingresos	lineasimpuesto	lineasticket	pagos	persona	productos	tickets	TABLE ACCES FULL	TABLE ACCES BY INDEX ROWID	INDEX RANGE SCAN	INDEX UNIQUE SCAN	HASH JOIN
Obtener un listado de los tickets realizados durante el año 2013 indicando código del ticket, nombre del cliente, nombre del producto, cantidad, precio, impuesto, fecha y el importe total debe ser entre 500 y 800 dólares, indicar también nombre de la persona que ha realizado la venta.	Simple	Nº Filas	2240000	10000000	10000000	29000000	112000	10	78763	10000000	8				7
		Costo	10343	8266	10165	35728	9486	2	208	8478					
		Nº Filas	2240000	20000000	20000000	59000000	190000	10	78763	20000000					
		Costo	10343	16569	20402	71550	19044	2	208	16990					
	Índice	Nº Filas	2240000	10000000	10000000	29000000	112000	10	78763	10000000	7	1	1		7
		Costo	10343	8266	10165	35728	243	2	208	8478					
		Nº Filas	2240000	20000000	20000000	59000000	190000	10	78763	20000000					
		Costo	10343	16569	20402	71550	410	2	208	16990					
	Subquery	Nº Filas	1	10000000	1	29000000	112000	1	1	10000000	5	3		3	3
		Costo	3	8266	10169	35728	9486	1	2	8478					
		Nº Filas	1	20000000	1	59000000	190000	1	1	20000000					
		Costo	3	16569	20411	71550	19044	1	2	16990					
	Join	Nº Filas	2240000	10000000	10000000	29000000	112000	10	78763	10000000	8				7
		Costo	10343	8266	10165	35728	9486	2	208	8478					
		Nº Filas	2240000	20000000	20000000	59000000	190000	10	78763	20000000					
		Costo	10343	16569	20402	71550	19044	2	208	16990					
	Subquery + Join	Nº Filas	1	10000000	10000000	29000000	112000	1	1	10000000	5	3		3	4
		Costo	3	8266	10165	35728	9486	1	2	8478					
		Nº Filas	1	20000000	20000000	59000000	190000	1	1	20000000					
		Costo	3	16569	20402	71550	19044	1	2	16990					
	Índice + join	Nº Filas	2240000	10000000	10000000	29000000	112000	10	78763	10000000	7	1	1		7
		Costo	10343	8266	10165	35728	243	2	208	8478					
		Nº Filas	2240000	20000000	20000000	59000000	190000	10	78763	20000000					
		Costo	10343	16569	20402	71550	410	2	208	16990					
	Índice + Subquery	Nº Filas	1	10000000	1	29000000	112000	1	1	10000000	4	4	1	3	3
		Costo	3	8266	10169	35728	243	1	2	8478					
		Nº Filas	1	20000000	1	59000000	190000	1	1	20000000					
		Costo	3	16569	20411	71550	410	1	2	16990					
	Índice + Subquery + Join	Nº Filas	1	10000000	10000000	29000000	112000	1	1	10000000	4	4	1	3	4
		Costo	3	8266	10165	35728	243	1	2	8478					
		Nº Filas	1	20000000	20000000	59000000	190000	1	1	20000000					
		Costo	3	16569	20402	71550	410	1	2	16990					

4.2 DISCUSIONES

Los resultados obtenidos demuestran que la aplicación de índices mejora notablemente los tiempos de respuestas de las consultas, concuerda con lo expresado por CORONEL, MORRIS, ROB (2013) quienes mencionan que si se crea índices en los atributos individuales utilizados en las condiciones de búsqueda, los DBMS accederán a la tabla mediante una exploración de índice en lugar de un escaneo completo de tabla obteniendo un menor tiempo de respuesta; en el manual de optimización desarrollado por el área de sistemas de la Universidad de Córdoba, Argentina mencionan que la la creación de un índice mejora notablemente la recuperación de datos; sin embargo, las operaciones de inserción y actualización de datos se verán penalizadas por el hecho de tener que actualizar también el índice. En sus pruebas establecen que un acceso indexado es más rápido que un acceso secuencial siempre y cuando la cantidad de datos no supere el 25% porque este porcentaje es buena estimación para el número de filas del cual resulta más rentable hacer una búsqueda secuencial .

Emplear inner join con millones de registros en ciertos casos es mejor que usar subquery el inner es más rápido en cuanto a tiempo de respuesta, aunque hay consultas que se ejecutan en un menor tiempo cuando se usa subquery todo depende en que DBMS se este trabajando, coincidiendo con el área de sistemas de la Universidad de Córdoba en su manual de optimización, quienes en sus pruebas de comparación entre join y subquery con 300.000 números de filas llegan a establecer que el uso del join es más eficiente que subquery aunque la diferencia de tiempo no es muy notable debido a la cantidad de datos que ellos emplearon para sus pruebas, no descartan la idea de usar subquery como alternativa al join. Hidalgo (2012) en su primera etapa de optimización de sus prueba menciona que el inner join es más rápido que emparejar tablas y registros en el where.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

En base a los resultados obtenidos en la presente investigación se pudo llegar a las siguientes conclusiones:

Al crear consultas combinadas para consultar datos de varias tablas la técnica más eficaz es la utilización del INNER JOIN explícito, con relación a la forma implícita con el WHERE el cual realiza una doble función con la unión de las tablas y la evaluación de las pruebas condicionales, con millones de registro se llega a la conclusión que el INNER JOIN es más rápido en 1% aunque el plan de ejecución es el mismo para ambos.

Usando subquery una buena estrategia de búsqueda es la utilización de esta en la lista de campos listados en el SELECT cuando se requiera utilizar la cláusula group by, la subquery permite que los campos se seleccionen de forma natural sin tener que incluirlos dentro del group by debido a que esta cláusula es costosa en tiempos de ejecución.

Los índices aumentan la velocidad de respuesta de las consultas en un 10% mejorando de esta manera su rendimiento y optimizando su resultado, debido a que los índices son la técnica más importante utilizada en la optimización del rendimiento de SQL, la clave está en saber cuándo usarlos.

5.2 RECOMENDACIONES

En consideración a los resultados obtenidos se recomienda:

Si se va a acceder con frecuencia a tablas con gran cantidad de registros según condiciones de selección o búsqueda en los valores de determinado campo, es conveniente crear un índice sobre el mismo, en cualquier DBMS que se utilice este presenta un mejor rendimiento de la consulta ya que en la mayor parte de los DBMS el índice es una estructura de acceso independiente, que se crea, reconstruye y elimina dinámicamente; en Oracle 11g un índice puede convertirse de forma inutilizable sin tener necesidad de eliminarlo. Es importante también cambiar el parámetro de configuración de índice asígnándole un tamaño inicial.

Al comparar rendimiento de consultas en SQL Server es útil limpiar la caché interna de memoria para probar las consultas en igualdad de condiciones, es decir evitar que una consulta parezca más rápida porque encuentra todas las páginas en caché, en lugar de ser una consulta realmente óptima.

Si utiliza Oracle es mejor hacer uso del inner join ya que es un gestor de bases de datos especialmente diseñado para resolver eficientemente los joins; por ello se recomienda usarlos en lugar de las subqueries.

CAPITULO VI

BIBLIOGRAFÍA

6.1 LITERATURA CITADA

Elmasri, R., & Navathe, S. (2010). Refinación de Consultas. Mexico: Pearson Educacion.

6.2 LINKOGRAFÍA

Alcalde, A. (21 de Julio de 2013). *El Baúl del Programador*. Obtenido de http://elbauldelprogramador.com/consulta-de-datos-subconsultas/

Alfaro, D. (15 de 12 de 2011). *Tunning en Bases de Datos Oracle y SQL Server*. Obtenido de http://blog.educacionit.com/2011/12/15/tunning-en-bases-de-datos-oracle-y-sql-server/

Casares, C. (S.f). Obtenido de http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/sql_tutorial.html

Escarrega, S. (s.f.). *slideshare.* Obtenido de http://es.slideshare.net/sergioescarregapalafox/tutorial-sql-38980974

Garavito, J. (Febrero de 2007). Obtenido de http://laboratorio.is.escuelaing.edu.co/labinfo/doc/Manual_Basico_del_Lenguaje_SQL.pdf

García, F. (19 de Julio de 2008). Obtenido de http://sqleficiente.wordpress.com/2008/07/

Grimpi IT Blog. (s.f.). Obtenido de https://grimpidev.wordpress.com/2008/10/28/entender-el-plan-de-ejecucion-en-sql-server-20052008/

Heroku dev center. (17 de Enero de 2014). Obtenido de https://devcenter.heroku.com/articles/postgresql-indexes

Herrarte, P. (22 de Octubre de 2005). Obtenido de http://www.devjoker.com/contenidos/catss/17/Consultas-combinadas-JOINS.aspx.

Herrarte, P. (22 de Octubre de 2005). Obtenido de http://www.devjoker.com/contenidos/articulos/12/Indices.aspx

Pilecki, M. (Noviembre de 2007). Obtenido de http://technet.microsoft.com/es-es/magazine/2007.11.sqlquery.aspx

Pullas, P. (20 de Enero de 2006). Obtenido de http://www.ecuoug.org/papers/PresentacionTips_20ENE2006.pdf

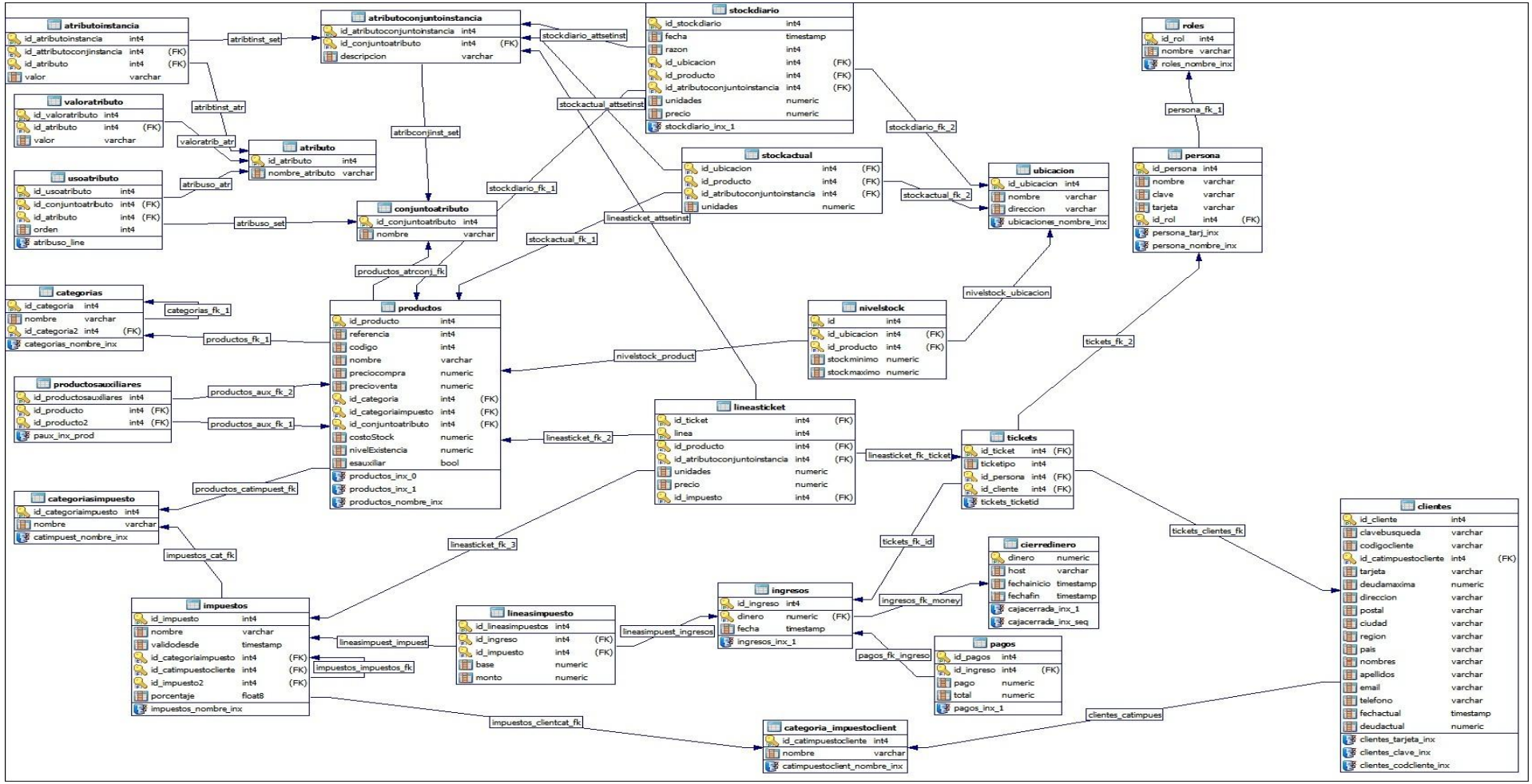
Sheldon, R. (25 de Noviembre de 2008). Obtenido de https://www.simple-talk.com/sql/learn-sql-server/sql-server-index-basics/

Winand, M. (14 de Julio de 2011). Obtenido de http://use-the-index-luke.com/sql/explain-plan/postgresql/operations

CAPITULO VII

ANEXOS

ANEXO 1. DIAGRAMA DE BASE DE DATOS



ANEXO 2

REQUERIMIENTOS PARA LAS CONSULTAS

CONSULTA 1: Se desea obtener un informe completo de las ventas realizadas por los cajeros en un rango de fecha y cuyo total de venta sean inferiores a los $500 dólares, de las ventas se debe mostrar lo siguiente: nombre de los productos, atributos, categoría, precio, iva; y de los cajeros se debe mostrar el monto.

CONSULTA 2: Mostrar código, nombre, dirección de los clientes que han comprado productos de la categoria farmacéuticos pertenecientes a los laboratorios: Quifatex Prop Merz, Naturpharma, Grunenthal-Genéricos, Vitafarma, Quifatex Genéricos. Mostrar el código del ticket, nombre del producto, precio de venta y las unidades vendidas.

CONSULTA 3: Mostrar un listado detallando la cantidad de unidades vendidas por productos que han sido registrados durante el segundo trimestre del año 2013, indicando código, precio de venta y descripción para los productos que se vendieron entre 600 y 1000 unidades, mostrar también el código y nombre de la categoría a la cual pertenece cada producto.

CONSULTA 4: Obtener un listado de los tickets realizados durante el año 2013 indicando código del ticket, nombre del cliente, nombre del producto, cantidad, precio, impuesto, fecha y el importe total debe ser entre 500 y 800 dólares, indicar también nombre de la persona que ha realizado la venta.

ANEXO 3

RESULTADOS EN POSTGRESQL 9.2 CON 10 MILLONES DE DATOS

	Consulta	Filas Devueltas	Tiempos de Respuestas (Segundos)	Volumen de Datos	Plan de Ejecución
Consulta 1	Simple	747432	242,57	10’	Ver Anexo 10
	Índice	747432	93,62	10’	Ver Anexo 11
	Subquery	747432	103,00	10’	Ver Anexo 12
	Join	747432	94,02	10’	Ver Anexo 10
	Subquery + Join	747432	97,2	10’	Ver Anexo 13
	Índice + Join	747432	133,29	10’	Ver Anexo 11
	Índice + Subquery	747432	100,05	10’	Ver Anexo 14
	Índice + Subquery + Join	747432	97,12	10’	Ver Anexo 15
Consulta 2	Simple	68148	29,92	10’	Ver Anexo 16
	Índice	68148	14,76	10’	Ver Anexo 17
	Subquery	68148	149,63	10’	Ver Anexo 18
	Join	68148	16,68	10’	Ver Anexo 19
	Subquery +Join	68148	149,91	10’	Ver Anexo 20
	Índice + Join	68148	14,65	10’	Ver Anexo 21
	Índice + Subquery	68148	77,55	10’	Ver Anexo 22
	Índice + Subquery + Join	68148	8,02	10’	Ver Anexo 23
Consulta 3	Simple	24156	351,54	10’	Ver Anexo 24
	Índice	24156	179,24	10’	Ver Anexo 25
	Subquery	24156	57,37	10’	Ver Anexo 26
	Join	24156	177,06	10’	Ver Anexo 24
	Subquery + Join	24156	236,26	10’	Ver Anexo 25
	Índice + Join	24156	327,38	10’	Ver Anexo 27
	Índice + Subquery	24156	61,07	10’	Ver Anexo 28
	Índice + Subquery + Join	24156	228,95	10’	Ver Anexo 29
Consulta 4	Simple	867960	304,75	10’	Ver Anexo 30
	Índice	867960	147,84	10’	Ver Anexo 31
	Subquery	867960	411,11	10’	Ver Anexo 32
	Join	867960	198,65	10’	Ver Anexo 30
	Subquery + Join	867960	236,11	10’	Ver Anexo 33
	Índice + Join	867960	206,85	10’	Ver Anexo 31
	Índice + Subquery	867960	334,02	10’	Ver Anexo 34
	Índice + Subquery + Join	867960	312,94	10’	Ver Anexo 35

ANEXO 4

RESULTADOS EN POSTGRESQL 9.2 CON 20 MILLONES DE DATOS

	Consulta	Filas Devueltas	Tiempos de Respuestas (Segundos)	Volumen de Datos	Plan de Ejecución
Consulta 1	Simple	1497576	537,86	20’	Ver Anexo 36
	Índice	1497576	457,25	20’	Ver Anexo 37
	Subquery	1497576	523,46	20’	Ver Anexo 38
	Join	1497576	474,7	20’	Ver Anexo 36
	Subquery + Join	1497576	656,9	20’	Ver Anexo 39
	Índice + Join	1497576	603,54	20’	Ver Anexo 37
	Índice + Subquery	1497576	606,48	20’	Ver Anexo 40
	Índice + Subquery + Join	1497576	539,89	20’	Ver Anexo 41
Consulta 2	Simple	135015	61,05	20’	Ver Anexo 42
	Índice	135015	60,32	20’	Ver Anexo 43
	Subquery	135015	394,13	20’	Ver Anexo 44
	Join	135015	56,85	20’	Ver Anexo 45
	Subconsultas +Join	135015	293,55	20’	Ver Anexo 46
	Índice + Join	135015	43,33	20’	Ver Anexo 47
	Índice + Subquery	135015	90,41	20’	Ver Anexo 48
	Índice + Subquery + Join	135015	15,94	20’	Ver Anexo 49
Consulta 3	Simple	2917	1139,22	20	Ver Anexo 50
	Índice	2917	871,11	20’	Ver Anexo 51
	Subquery	2917	700,25	20’	Ver Anexo 52
	Join	2917	702,54	20’	Ver Anexo 50
	Subquery + Join	2917	1040,72	20’	Ver Anexo 53
	Índice + Join	2917	927,79	20’	Ver Anexo 51
	Índice + Subquery	2917	756,62	20’	Ver Anexo 54
	Índice + Subquery + Join	2917	952,17	20’	Ver Anexo 55
Consulta 4	Simple	1733012	822,23	20’	Ver Anexo 56
	Índice	1733012	765,29	20’	Ver Anexo 57
	Subquery	1733012	1432,51	20’	Ver Anexo 58
	Join	1733012	890,59	20’	Ver Anexo 56
	Subquery + Join	1733012	1144,11	20’	Ver Anexo 59
	Índice + Join	1733012	721,65	20’	Ver Anexo 57
	Índice + Subquery	1733012	1355,12	20’	Ver Anexo 60
	Índice + Subquery + Join	1733012	1295,22	20’	Ver Anexo 61

ANEXO 5

RESULTADOS EN SQL SERVER R2 2008 CON 10 MILLONES DE DATOS

	Consulta	Filas Devueltas	Tiempos de Respuestas (Segundos)	Volumen de Datos	Plan de Ejecución
Consulta 1	Simple	747432	61,2	10’	Ver Anexo 62
	Índice	747432	55,0 s	10’	Ver Anexo 63
	Subquery	747432	67,2	10’	Ver Anexo 64
	Join	747432	62,4	10’	Ver Anexo 62
	Subquery + Join	747432	61,2	10’	Ver Anexo 65
	Índice + Join	747432	55,0	10’	Ver Anexo 66
	Índice + Subquery	747432	58,0	10’	Ver Anexo 67
	Índice + Subquery + Join	747432	57,0	10’	Ver Anexo 68
Consulta 2	Simple	68148	36,0	10’	Ver Anexo 69
	Índice	68148	15,0	10’	Ver Anexo 70
	Subquery	68148	37,0	10’	Ver Anexo 71
	Join	68148	35,0	10’	Ver Anexo 69
	Subquery +Join	68148	36,0	10’	Ver Anexo 72
	Índice + Join	68148	12,0	10’	Ver Anexo 70
	Índice + Subquery	68148	13,0	10’	Ver Anexo 73
	Índice + Subquery + Join	68148	12,0	10’	Ver Anexo 74
Consulta 3	Simple	24156	121,2	10’	Ver Anexo 75
	Índice	24156	120,6	10’	Ver Anexo 76
	Subquery	24156	31,0	10’	Ver Anexo 77
	Join	24156	123,0	10’	Ver Anexo 75
	Subquery + Join	24156	29,0	10’	Ver Anexo 78
	Índice + Join	24156	120,0	10’	Ver Anexo 76
	Índice + Subquery	24156	26,0	10’	Ver Anexo 79
	Índice + Subquery + Join	24156	25,0	10’	Ver Anexo 80
Consulta 4	Simple	867960	80,4	10’	Ver Anexo 81
	Índice	867960	78,6	10’	Ver Anexo 82
	Subquery	867960	67,2	10’	Ver Anexo 83
	Join	867960	78,6	10’	Ver Anexo 84
	Subquery + Join	867960	58,0	10’	Ver Anexo 85
	Índice + Join	867960	76,8	10’	Ver Anexo 86
	Índice + Subquery	867960	61,8	10’	Ver Anexo 87
	Índice + Subquery + Join	867960	47,0	10’	Ver Anexo 88

ANEXO 6

RESULTADOS EN SQL SERVER R2 2008 CON 20 MILLONES DE DATOS

	Consulta	Filas Devueltas	Tiempos de Respuestas (Segundos)	Volumen de Datos	Plan de Ejecución
Consulta 1	Simple	1497576	129,0	20’	Ver Anexo 89
	Índice	1497576	78,0	20’	Ver Anexo 90
	Subquery	1497576	132,0	20’	Ver Anexo 91
	Join	1497576	122,4	20’	Ver Anexo 89
	Subquery + Join	1497576	127,2	20’	Ver Anexo 92
	Índice + Join	1497576	78,0	20’	Ver Anexo 90
	Índice + Subquery	1497576	80,4	20’	Ver Anexo 93
	Índice + Subquery + Join	1497576	75,0	20’	Ver Anexo 94
Consulta 2	Simple	135015	84,0	20’	Ver Anexo 95
	Índice	135015	27,0	20’	Ver Anexo 96
	Subquery	135015	80,4	20’	Ver Anexo 97
	Join	135015	80,4	20’	Ver Anexo 95
	Subconsultas +Join	135015	79,8	20’	Ver Anexo 98
	Índice + Join	135015	26,0	20’	Ver Anexo 96
	Índice + Subquery	135015	25,0	20’	Ver Anexo 99
	Índice + Subquery + Join	135015	26,0	20’	Ver Anexo 100
Consulta 3	Simple	2917	273,0	20	Ver Anexo 101
	Índice	2917	258,6	20’	Ver Anexo 102
	Subquery	2917	78,0	20’	Ver Anexo 103
	Join	2917	268,8	20’	Ver Anexo 101
	Subquery + Join	2917	81,0	20’	Ver Anexo 104
	Índice + Join	2917	261,6	20’	Ver Anexo 102
	Índice + Subquery	2917	71,0	20’	Ver Anexo 105
	Índice + Subquery + Join	2917	70,2	20’	Ver Anexo 106
Consulta 4	Simple	1733012	394,2	20’	Ver Anexo 107
	Índice	1733012	181,4	20’	Ver Anexo 108
	Subquery	1733012	188,4	20’	Ver Anexo 109
	Join	1733012	374,4	20’	Ver Anexo 107
	Subquery + Join	1733012	360,0	20’	Ver Anexo 110
	Índice + Join	1733012	191,4	20’	Ver Anexo 108
	Índice + Subquery	1733012	149,4	20’	Ver Anexo 111
	Índice + Subquery + Join	1733012	246,0	20’	Ver Anexo 112

ANEXO 7

RESULTADOS EN ORACLE 11g R2 CON 10 MILLONES DE DATOS

	Consulta	Filas Devueltas	Tiempos de Respuestas (Segundos)	Volumen de Datos	Plan de Ejecución
Consulta 1	Simple	747432	49,23	10’	Ver Anexo 113
	Índice	747432	42,62	10’	Ver Anexo 114
	Subquery	747432	100,52	10’	Ver Anexo 115
	Join	747432	45,88	10’	Ver Anexo 116
	Subquery + Join	747432	52,45	10’	Ver Anexo 117
	Índice + Join	747432	41,26	10’	Ver Anexo 118
	Índice + Subquery	747432	90,35	10’	Ver Anexo 119
	Índice + Subquery + Join	747432	46,61	10’	Ver Anexo 120
Consulta 2	Simple	68148	25,07	10’	Ver Anexo 121
	Índice	68148	8,79	10’	Ver Anexo 122
	Subquery	68148	23,28	10’	Ver Anexo 123
	Join	68148	22,62	10’	Ver Anexo 121
	Subquery +Join	68148	23,27	10’	Ver Anexo 124
	Índice + Join	68148	8,18	10’	Ver Anexo 122
	Índice + Subquery	68148	8,63	10’	Ver Anexo 125
	Índice + Subquery + Join	68148	8,32	10’	Ver Anexo 126
Consulta 3	Simple	24156	44,62	10’	Ver Anexo 127
	Índice	24156	37,47	10’	Ver Anexo 128
	Subquery	24156	38,79	10’	Ver Anexo 129
	Join	24156	45,17	10’	Ver Anexo 127
	Subquery + Join	24156	41,91	10’	Ver Anexo 130
	Índice + Join	24156	34,21	10’	Ver Anexo 128
	Índice + Subquery	24156	32,56	10’	Ver Anexo 131
	Índice + Subquery + Join	24156	34,54	10’	Ver Anexo 132
Consulta 4	Simple	867960	98,22	10’	Ver Anexo 133
	Índice	867960	65,76	10’	Ver Anexo 134
	Subquery	867960	100,13	10’	Ver Anexo 135
	Join	867960	97,02	10’	Ver Anexo 133
	Subquery + Join	867960	559,28	10’	Ver Anexo 136
	Índice + Join	867960	95,49	10’	Ver Anexo 134
	Índice + Subquery	867960	101,14	10’	Ver Anexo 137
	Índice + Subquery + Join	867960	563,22	10’	Ver Anexo 138

ANEXO 8

RESULTADOS EN ORACLE 11g R2 CON 20 MILLONES DE DATOS

	Consulta	Filas Devueltas	Tiempos de Respuestas (Segundos)	Volumen de Datos	Plan de Ejecución
Consulta 1	Simple	1497576	254,10	20’	Ver Anexo 139
	Índice	1497576	210,05	20’	Ver Anexo 140
	Subquery	1497576	450,91	20’	Ver Anexo 141
	Join	1497576	233,14	20’	Ver Anexo 139
	Subquery + Join	1497576	301,28	20’	Ver Anexo 142
	Índice + Join	1497576	202,28	20’	Ver Anexo 140
	Índice + Subquery	1497576	435,06	20’	Ver Anexo 143
	Índice + Subquery + Join	1497576	238,26	20’	Ver Anexo 144
Consulta 2	Simple	135015	40,52	20’	Ver Anexo 145
	Índice	135015	169,58	20’	Ver Anexo 146
	Subquery	135015	40,59	20’	Ver Anexo 147
	Join	135015	41,44	20’	Ver Anexo 145
	Subconsultas +Join	135015	41,39	20’	Ver Anexo 148
	Índice + Join	135015	155,73	20’	Ver Anexo 146
	Índice + Subquery	135015	154,24	20’	Ver Anexo 149
	Índice + Subquery + Join	135015	187,01	20’	Ver Anexo 150
Consulta 3	Simple	2917	150,39	20	Ver Anexo 151
	Índice	2917	106,97	20’	Ver Anexo 152
	Subquery	2917	73,08	20’	Ver Anexo 153
	Join	2917	120,47	20’	Ver Anexo 154
	Subquery + Join	2917	102,27	20’	Ver Anexo 155
	Índice + Join	2917	107,88	20’	Ver Anexo 152
	Índice + Subquery	2917	89,43	20’	Ver Anexo 156
	Índice + Subquery + Join	2917	79,45	20’	Ver Anexo 157
Consulta 4	Simple	1733012	350,22	20’	Ver Anexo 158
	Índice	1733012	310,21	20’	Ver Anexo 159
	Subquery	1733012	400,51	20’	Ver Anexo 160
	Join	1733012	360,20	20’	Ver Anexo 158
	Subquery + Join	1733012	1201,02	20’	Ver Anexo 161
	Índice + Join	1733012	340,01	20’	Ver Anexo 159
	Índice + Subquery	1733012	410,20	20’	Ver Anexo 162
	Índice + Subquery + Join	1733012	1243,14	20’	Ver Anexo 163

ANEXO 9

TIEMPOS DE RESPUESTA EN LOS DIFERENTES GESTORES DE BASE DE DATOS

Descripción	Prueba	Consulta	Tiempo de Respuesta (segundos)					
			PostgreSQL 9.2		SQL Server 2008 R2		Oracle 11g	
			10 millones de datos	20 millones de datos	10 millones de datos	20 millones de datos	10 millones de datos	20 millones de datos
Se desea obtener un informe completo de las ventas realizadas por los cajeros en un rango de fecha y cuyo total de venta sean inferiores a los $500 dólares, de las ventas se debe mostrar lo siguiente: nombre de los productos, atributos, categoría, precio, iva; y de los cajeros se debe mostrar el monto.	1	Simple	242,57	537,86	61,2	129	49,23	254,10
	2	Índice	93,62	457,25	55	78	42,62	210,05
	3	Subquery	103	523,46	67,2	132	100,52	450,91
	4	Join	94,02	474,7	62,4	122,4	45,88	233,14
	5	Subquery + Join	97,2	656,9	61,2	127,2	52,45	301,28
	6	Índice + join	133,29	603,54	55	78	41,26	202,28
	7	Índice + Subquery	100,05	606,48	58	80,4	90,35	435,06
	8	Índice + Subquery + Join	97,12	539,89	57	75	46,61	238,26
Mostrar código, nombre, dirección de los clientes que han comprado productos de la categoria farmacéuticos pertenecientes a los laboratorios: QUIFATEX PROP MERZ, NATURPHARMA, GRUNENTHAL-GENERICOS, VITAFARMA, QUIFATEX GENERICOS. Mostrar el código del ticket, nombre del producto , precio de venta y las unidades vendidas.	1	Simple	29,92	61,05	36	84	25,07	40,52
	2	Índice	14,76	60,32	15	27	8,79	169,58
	3	Subquery	149,63	394,15	37	80,4	23,28	40,59
	4	Join	16,68	56,85	35	80,4	22,62	41,44
	5	Subquery + Join	149,91	293,55	36	79,8	23,27	41,39
	6	Índice + join	14,65	43,33	12	26	8,18	155,73
	7	Índice + Subquery	77,55	90,41	13	25	8,63	154,24
	8	Índice + Subquery + Join	8,02	15,94	12	26	8,32	187,01
Mostrar un listado detallando la cantidad de unidades vendidas por productos que han sido registrados durante el segundo trimestre del año 2013, indicando código, precio de venta y descripción para los productos que se vendieron entre 600 y 1000 unidades, mostrar también el código y nombre de la categoría a la cual pertenece cada producto.	1	Simple	351,54	1139,22	121,2	273	44,62	150,39
	2	Índice	179,24	871,11	120,6	258,6	37,47	106,97
	3	Subquery	57,37	700,25	31	78	38,79	73,08
	4	Join	177,06	702,54	123	268,8	45,17	120,47
	5	Subquery + Join	236,26	1040,72	29	81	41,91	102,27
	6	Índice + join	327,38	927,79	120	261,6	34,21	107,88
	7	Índice + Subquery	61,07	756,62	26	71	32,56	89,43
	8	Índice + Subquery + Join	228,95	952,17	25	70,2	34,54	79,45
Obtener un listado de los tickets realizados durante el año 2013 indicando código del ticket, nombre del cliente, nombre del producto, cantidad, precio, impuesto, fecha y el importe total debe ser entre 500 y 800 dólares, indicar también nombre de la persona que ha realizado la venta.	1	Simple	304,75	822,23	80,4	394,2	98,22	350,22
	2	Índice	147,84	765,29	78,6	181,4	65,76	310,21
	3	Subquery	411,11	1432,51	67,2	188,4	100,13	400,51
	4	Join	198,65	890,59	78,6	374,4	97,02	360,20
	5	Subquery + Join	236,11	1144,11	58	360	559,28	1201,02
	6	Índice + join	206,85	721,65	76,8	191,4	95,49	340,01
	7	Índice + Subquery	334,02	1355,12	61,8	149,4	101,14	410,20
	8	Índice + Subquery + Join	312,94	1295,22	47	246	563,22	1243,14

ANEXO 10

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 1 SELECT SIMPLE Y SELECT USANDO JOIN 10 MILLONES DE DATOS.

```
"Hash Join  (cost=780345.94..1781347.30 rows=248205 width=86)"
"  Hash Cond: (l.id_atributoconjuntoinstancia = a.id_atributoconjuntoinstancia)"
"  ->  Hash Join  (cost=780311.06..1776658.57 rows=248205 width=53)"
"        Hash Cond: (l.id_producto = p.id_producto)"
"        ->  Hash Join  (cost=776992.89..1764645.23 rows=253042 width=27)"
"              Hash Cond: (l.id_ticket = i.id_ingreso)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=21)"
"              ->  Hash  (cost=775443.53..775443.53 rows=84349 width=18)"
"                    ->  Hash Join  (cost=450750.21..775443.53 rows=84349 width=18)"
"                          Hash Cond: (im.id_ingreso = i.id_ingreso)"
"                          ->  Seq Scan on lineasimpuesto im  (cost=0.00..188704.00 rows=9974621 width=10)"
"                                Filter: (monto < 500::numeric)"
"                          ->  Hash  (cost=449362.16..449362.16 rows=84564 width=8)"
"                                ->  Hash Join  (cost=217964.21..449362.16 rows=84564 width=8)"
"                                      Hash Cond: (i.id_ingreso = t.id_ticket)"
"                                      ->  Seq Scan on ingresos i  (cost=0.00..213695.00 rows=845641 width=4)"
"                                            Filter: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"                                      ->  Hash  (cost=201557.21..201557.21 rows=1000000 width=4)"
"                                            ->  Hash Join  (cost=2.21..201557.21 rows=1000000 width=4)"
"                                                  Hash Cond: (t.id_persona = ps.id_persona)"
"                                                  ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=8)"
"                                                  ->  Hash  (cost=2.20..2.20 rows=1 width=4)"
"                                                        ->  Hash Join  (cost=1.05..2.20 rows=1 width=4)"
"                                                              Hash Cond: (ps.id_rol = r.id_rol)"
"                                                              ->  Seq Scan on persona ps  (cost=0.00..1.10 rows=10 width=8)"
"                                                              ->  Hash  (cost=1.04..1.04 rows=1 width=4)"
"                                                                    ->  Seq Scan on roles r  (cost=0.00..1.04 rows=1 width=4)"
"                                                                          Filter: ((nombre)::text = 'Dependiente'::text)"
"        ->  Hash  (cost=1794.63..1794.63 rows=78763 width=30)"
"              ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=30)"
"  ->  Hash  (cost=21.06..21.06 rows=1106 width=37)"
"        ->  Seq Scan on atributoconjuntoinstancia a  (cost=0.00..21.06 rows=1106 width=37)"
```

ANEXO 11

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 1 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 10 MILLONES DE DATOS.

```
"Hash Join  (cost=659015.22..1660034.31 rows=248492 width=86)"
"  Hash Cond: (l.id_atributoconjuntoinstancia = a.id_atributoconjuntoinstancia)"
"  ->  Hash Join  (cost=658980.34..1655340.20 rows=248492 width=53)"
"        Hash Cond: (l.id_producto = p.id_producto)"
"        ->  Hash Join  (cost=655662.17..1643317.45 rows=253334 width=27)"
"              Hash Cond: (l.id_ticket = i.id_ingreso)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=21)"
"              ->  Hash  (cost=654111.58..654111.58 rows=84447 width=18)"
"                    ->  Hash Join  (cost=329417.28..654111.58 rows=84447 width=18)"
"                          Hash Cond: (im.id_ingreso = i.id_ingreso)"
"                          ->  Seq Scan on lineasimpuesto im  (cost=0.00..188704.00 rows=9974621 width=10)"
"                                Filter: (monto < 500::numeric)"
"                          ->  Hash  (cost=328028.01..328028.01 rows=84662 width=8)"
"                                ->  Hash Join  (cost=108254.39..328028.01 rows=84662 width=8)"
"                                      Hash Cond: (t.id_ticket = i.id_ingreso)"
"                                      ->  Hash Join  (cost=2.21..201557.21 rows=1000000 width=4)"
"                                            Hash Cond: (t.id_persona = ps.id_persona)"
"                                            ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=8)"
"                                            ->  Hash  (cost=2.20..2.20 rows=1 width=4)"
"                                                  ->  Hash Join  (cost=1.05..2.20 rows=1 width=4)"
"                                                        Hash Cond: (ps.id_rol = r.id_rol)"
"                                                        ->  Seq Scan on persona ps  (cost=0.00..1.10 rows=10 width=8)"
"                                                        ->  Hash  (cost=1.04..1.04 rows=1 width=4)"
"                                                              ->  Seq Scan on roles r  (cost=0.00..1.04 rows=1 width=4)"
"                                                                    Filter: ((nombre)::text = 'Dependiente'::text)"
"                                      ->  Hash  (cost=94361.45..94361.45 rows=846618 width=4)"
"                                            ->  Bitmap Heap Scan on ingresos i  (cost=17967.18..94361.45 rows=846618 width=4)"
"                                                  Recheck Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"                                                  ->  Bitmap Index Scan on ingresos_inx_1  (cost=0.00..17755.53 rows=846618 width=0)"
"                                                        Index Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"        ->  Hash  (cost=1794.63..1794.63 rows=78763 width=30)"
"              ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=30)"
"  ->  Hash  (cost=21.06..21.06 rows=1106 width=37)"
"        ->  Seq Scan on atributoconjuntoinstancia a  (cost=0.00..21.06 rows=1106 width=37)"
```

ANEXO 12

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 1 SELECT USANDO SUBQUERY 10 MILLONES DE DATOS.

```
"Hash Semi Join  (cost=1262268.79..30632720.17 rows=1711586 width=21)"
"  Hash Cond: (lineasimpuesto.id_ingreso = public.tickets.id_ticket)"
"  ->  Hash Semi Join  (cost=551302.98..1546906.07 rows=2536861 width=29)"
"        Hash Cond: (l.id_ticket = lineasimpuesto.id_ingreso)"
"        ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=21)"
"        ->  Hash  (cost=537464.29..537464.29 rows=843495 width=8)"
"              ->  Hash Semi Join  (cost=227569.51..537464.29 rows=843495 width=8)"
"                    Hash Cond: (lineasimpuesto.id_ingreso = public.ingresos.id_ingreso)"
"                    ->  Seq Scan on lineasimpuesto  (cost=0.00..188704.00 rows=9974621 width=4)"
"                          Filter: (monto < 500::numeric)"
"                    ->  Hash  (cost=213695.00..213695.00 rows=845641 width=4)"
"                          ->  Seq Scan on ingresos  (cost=0.00..213695.00 rows=845641 width=4)"
"                                Filter: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"  ->  Hash  (cost=694558.81..694558.81 rows=1000000 width=4)"
"        ->  Hash Semi Join  (cost=503128.81..694558.81 rows=1000000 width=4)"
"              Hash Cond: (public.tickets.id_persona = persona.id_persona)"
"              ->  Seq Scan on tickets  (cost=0.00..154055.00 rows=10000000 width=8)"
"              ->  Hash  (cost=503128.80..503128.80 rows=1 width=8)"
"                    ->  Nested Loop Semi Join  (cost=227569.51..503128.80 rows=1 width=8)"
"                          Join Filter: (persona.id_persona = public.tickets.id_persona)"
"                          ->  Nested Loop Semi Join  (cost=0.00..13.59 rows=1 width=4)"
"                                Join Filter: (persona.id_rol = roles.id_rol)"
"                                ->  Index Scan using persona_pkey on persona  (cost=0.00..12.40 rows=10 width=8)"
"                                ->  Materialize  (cost=0.00..1.04 rows=1 width=4)"
"                                      ->  Seq Scan on roles  (cost=0.00..1.04 rows=1 width=4)"
"                                            Filter: ((nombre)::text = 'Dependiente'::text)"
"                          ->  Hash Semi Join  (cost=227569.51..503114.96 rows=845641 width=4)"
"                                Hash Cond: (public.tickets.id_ticket = public.ingresos.id_ingreso)"
"                                ->  Seq Scan on tickets  (cost=0.00..154055.00 rows=10000000 width=8)"
"                                ->  Hash  (cost=213695.00..213695.00 rows=845641 width=4)"
"                                      ->  Seq Scan on ingresos  (cost=0.00..213695.00 rows=845641 width=4)"
"                                            Filter: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Index Scan using atribconjuninst_pkey on atributoconjuntoinstancia a  (cost=0.00..8.27 rows=1 width=33)"
"          Index Cond: (id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia)"
```

ANEXO 13

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 1 SELECT USANDO SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
"Hash Join  (cost=897954.41..22873147.34 rows=1265212 width=27)"
"  Hash Cond: (l.id_ticket = i.id_ingreso)"
"  ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=21)"
"  ->  Hash  (cost=890210.56..890210.56 rows=421748 width=18)"
"        ->  Hash Join  (cost=548353.97..890210.56 rows=421748 width=18)"
"              Hash Cond: (im.id_ingreso = i.id_ingreso)"
"              ->  Seq Scan on lineasimpuesto im  (cost=0.00..188704.00 rows=9974621 width=10)"
"                    Filter: (monto < 500::numeric)"
"              ->  Hash  (cost=541416.71..541416.71 rows=422821 width=8)"
"                    ->  Hash Join  (cost=227575.94..541416.71 rows=422821 width=8)"
"                          Hash Cond: (t.id_persona = ps.id_persona)"
"                          ->  Hash Join  (cost=227569.51..534010.92 rows=845641 width=12)"
"                                Hash Cond: (t.id_ticket = i.id_ingreso)"
"                                ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=8)"
"                                ->  Hash  (cost=213695.00..213695.00 rows=845641 width=4)"
"                                      ->  Seq Scan on ingresos i  (cost=0.00..213695.00 rows=845641 width=4)"
"                                            Filter: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"                          ->  Hash  (cost=6.36..6.36 rows=5 width=4)"
"                                ->  Seq Scan on persona ps  (cost=0.00..6.36 rows=5 width=4)"
"                                      Filter: (SubPlan 3)"
"                                      SubPlan 3"
"                                        ->  Seq Scan on roles r  (cost=0.00..1.04 rows=1 width=4)"
"                                              Filter: ((ps.id_rol = id_rol) AND ((nombre)::text = 'Dependiente'::text))"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Index Scan using atribconjuninst_pkey on atributoconjuntoinstancia a  (cost=0.00..8.27 rows=1 width=33)"
"          Index Cond: (id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia)"
```

ANEXO 14

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 1 SELECT CON ÍNDICE + SUBQUERY 10 MILLONES DE DATOS.

```
"Hash Semi Join  (cost=1023684.39..30426949.30 rows=1713563 width=21)"
"  Hash Cond: (lineasimpuesto.id_ingreso = public.tickets.id_ticket)"
"  ->  Hash Semi Join  (cost=432018.85..1427665.14 rows=2539792 width=29)"
"        Hash Cond: (l.id_ticket = lineasimpuesto.id_ingreso)"
"        ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=21)"
"        ->  Hash  (cost=418163.98..418163.98 rows=844469 width=8)"
"              ->  Hash Semi Join  (cost=108252.18..418163.98 rows=844469 width=8)"
"                    Hash Cond: (lineasimpuesto.id_ingreso = public.ingresos.id_ingreso)"
"                    ->  Seq Scan on lineasimpuesto  (cost=0.00..188704.00 rows=9974621 width=4)"
"                          Filter: (monto < 500::numeric)"
"                    ->  Hash  (cost=94361.45..94361.45 rows=846618 width=4)"
"                          ->  Bitmap Heap Scan on ingresos  (cost=17967.18..94361.45 rows=846618 width=4)"
"                                Recheck Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"                                ->  Bitmap Index Scan on ingresos_inx_1  (cost=0.00..17755.53 rows=846618 width=0)"
"                                      Index Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"  ->  Hash  (cost=575258.54..575258.54 rows=1000000 width=4)"
"        ->  Hash Semi Join  (cost=383828.54..575258.54 rows=1000000 width=4)"
"              Hash Cond: (public.tickets.id_persona = persona.id_persona)"
"              ->  Seq Scan on tickets  (cost=0.00..154055.00 rows=10000000 width=8)"
"              ->  Hash  (cost=383828.53..383828.53 rows=1 width=8)"
"                    ->  Nested Loop Semi Join  (cost=108252.18..383828.53 rows=1 width=8)"
"                          Join Filter: (persona.id_persona = public.tickets.id_persona)"
"                          ->  Nested Loop Semi Join  (cost=0.00..13.59 rows=1 width=4)"
"                                Join Filter: (persona.id_rol = roles.id_rol)"
"                                ->  Index Scan using persona_pkey on persona  (cost=0.00..12.40 rows=10 width=8)"
"                                ->  Materialize  (cost=0.00..1.04 rows=1 width=4)"
"                                      ->  Seq Scan on roles  (cost=0.00..1.04 rows=1 width=4)"
"                                            Filter: ((nombre)::text = 'Dependiente'::text)"
"                          ->  Hash Semi Join  (cost=108252.18..383814.69 rows=846618 width=4)"
"                                Hash Cond: (public.tickets.id_ticket = public.ingresos.id_ingreso)"
"                                ->  Seq Scan on tickets  (cost=0.00..154055.00 rows=10000000 width=8)"
"                                ->  Hash  (cost=94361.45..94361.45 rows=846618 width=4)"
"                                      ->  Bitmap Heap Scan on ingresos  (cost=17967.18..94361.45 rows=846618 width=4)"
"                                            Recheck Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"                                            ->  Bitmap Index Scan on ingresos_inx_1  (cost=0.00..17755.53 rows=846618 width=0)"
"                                                  Index Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Index Scan using atribconjuninst_pkey on atributoconjuntoinstancia a  (cost=0.00..8.27 rows=1 width=33)"
"          Index Cond: (id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia)"
```

ANEXO 15

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 1 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
"Hash Join  (cost=778683.44..22778072.01 rows=1266673 width=27)"
"  Hash Cond: (l.id_ticket = i.id_ingreso)"
"  ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=21)"
"  ->  Hash  (cost=770930.51..770930.51 rows=422235 width=18)"
"        ->  Hash Join  (cost=429067.05..770930.51 rows=422235 width=18)"
"              Hash Cond: (im.id_ingreso = i.id_ingreso)"
"              ->  Seq Scan on lineasimpuesto im  (cost=0.00..188704.00 rows=9974621 width=10)"
"                    Filter: (monto < 500::numeric)"
"              ->  Hash  (cost=422121.69..422121.69 rows=423309 width=8)"
"                    ->  Hash Join  (cost=108258.60..422121.69 rows=423309 width=8)"
"                          Hash Cond: (t.id_persona = ps.id_persona)"
"                          ->  Hash Join  (cost=108252.18..414707.36 rows=846618 width=12)"
"                                Hash Cond: (t.id_ticket = i.id_ingreso)"
"                                ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=8)"
"                                ->  Hash  (cost=94361.45..94361.45 rows=846618 width=4)"
"                                      ->  Bitmap Heap Scan on ingresos i  (cost=17967.18..94361.45 rows=846618 width=4)"
"                                            Recheck Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"                                            ->  Bitmap Index Scan on ingresos_inx_1  (cost=0.00..17755.53 rows=846618 width=0)"
"                                                  Index Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"                          ->  Hash  (cost=6.36..6.36 rows=5 width=4)"
"                                ->  Seq Scan on persona ps  (cost=0.00..6.36 rows=5 width=4)"
"                                      Filter: (SubPlan 3)"
"                                      SubPlan 3"
"                                        ->  Seq Scan on roles r  (cost=0.00..1.04 rows=1 width=4)"
"                                              Filter: ((ps.id_rol = id_rol) AND ((nombre)::text = 'Dependiente'::text))"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Index Scan using atribconjuninst_pkey on atributoconjuntoinstancia a  (cost=0.00..8.27 rows=1 width=33)"
"          Index Cond: (id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia)"
```

ANEXO 16

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT SIMPLE 10 MILLONES DE DATOS.

```
"Merge Join  (cost=966455.12..1074862.02 rows=66515 width=176)"
"  Merge Cond: (cl.id_cliente = t.id_cliente)"
"  ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..101640.91 rows=2240818 width=44)"
"  ->  Materialize  (cost=966455.09..966787.66 rows=66515 width=136)"
"        ->  Sort  (cost=966455.09..966621.38 rows=66515 width=136)"
"              Sort Key: t.id_cliente"
"              ->  Hash Join  (cost=647430.62..956576.77 rows=66515 width=136)"
"                    Hash Cond: (t.id_ticket = l.id_ticket)"
"                    ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=8)"
"                    ->  Hash  (cost=645299.19..645299.19 rows=66515 width=132)"
"                          ->  Nested Loop  (cost=3423.20..645299.19 rows=66515 width=132)"
"                                Join Filter: (p.id_categoria = c.id_categoria)"
"                                ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=1 width=52)"
"                                      Filter: ((nombre)::text = 'FARMACEUTICOS'::text)"
"                                ->  Hash Join  (cost=3423.20..643635.31 rows=133028 width=84)"
"                                      Hash Cond: (l.id_producto = p.id_producto)"
"                                      ->  Hash Join  (cost=28.04..634457.18 rows=135621 width=50)"
"                                            Hash Cond: (l.id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                                            ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=17)"
"                                            ->  Hash  (cost=27.97..27.97 rows=5 width=37)"
"                                                  ->  Seq Scan on atributoconjuntoinstancia aci  (cost=0.00..27.97 rows=5 width=37)"
"                                                        Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO
VITAFARMA","ATRIB.CONJUNTO QUIFATEX GENERICOS"}'::text[]))"
"                                      ->  Hash  (cost=1794.63..1794.63 rows=78763 width=38)"
"                                            ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=38)"
```

ANEXO 17

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT USANDO ÍNDICE 10 MILLONES DE DATOS.

```
"Hash Join  (cost=716678.85..840670.31 rows=66515 width=176)"
"  Hash Cond: (cl.id_cliente = t.id_cliente)"
"  ->  Seq Scan on clientes cl  (cost=0.00..65830.18 rows=2240818 width=44)"
"  ->  Hash  (cost=714547.41..714547.41 rows=66515 width=136)"
"        ->  Hash Join  (cost=405401.26..714547.41 rows=66515 width=136)"
"              Hash Cond: (t.id_ticket = l.id_ticket)"
"              ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=8)"
"              ->  Hash  (cost=403269.82..403269.82 rows=66515 width=132)"
"                    ->  Nested Loop  (cost=3908.92..403269.82 rows=66515 width=132)"
"                          Join Filter: (p.id_categoria = c.id_categoria)"
"                          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=1 width=52)"
"                                Filter: ((nombre)::text = 'FARMACEUTICOS'::text)"
"                          ->  Hash Join  (cost=3908.92..401605.95 rows=133028 width=84)"
"                                Hash Cond: (l.id_producto = p.id_producto)"
"                                ->  Nested Loop  (cost=513.75..392427.82 rows=135621 width=50)"
"                                      ->  Seq Scan on atributoconjuntoinstancia aci  (cost=0.00..27.97 rows=5 width=37)"
"                                            Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO
VITAFARMA","ATRIB.CONJUNTO QUIFATEX GENERICOS"}'::text[]))"
"                                      ->  Bitmap Heap Scan on lineasticket l  (cost=513.75..78208.73 rows=27124 width=17)"
"                                            Recheck Cond: (id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                                            ->  Bitmap Index Scan on atribconjinst_descrip  (cost=0.00..506.97 rows=27124 width=0)"
"                                                  Index Cond: (id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                                ->  Hash  (cost=1794.63..1794.63 rows=78763 width=38)"
"                                      ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=38)"
```

ANEXO 18

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT USANDO SUBQUERY 10 MILLONES DE DATOS.

```
"Nested Loop  (cost=0.00..280276995.75 rows=678 width=61)"
"  ->  Nested Loop  (cost=0.00..280242730.34 rows=678 width=21)"
"        ->  Nested Loop Semi Join  (cost=0.00..280236628.40 rows=678 width=17)"
"              Join Filter: (l.id_atributoconjuntoinstancia = a.id_atributoconjuntoinstancia)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..280225350.72 rows=149996 width=17)"
"                    Filter: (((SubPlan 6))::text = 'FARMACEUTICOS'::text)"
"                    SubPlan 6"
"                      ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"                            Join Filter: (p.id_categoria = c.id_categoria)"
"                            ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                                  Index Cond: (id_producto = l.id_producto)"
"                            ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"              ->  Materialize  (cost=0.00..28.00 rows=5 width=4)"
"                    ->  Seq Scan on atributoconjuntoinstancia a  (cost=0.00..27.97 rows=5 width=4)"
"                          Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO VITAFARMA","ATRIB.CONJUNTO
QUIFATEX GENERICOS"}'::text[]))"
"        ->  Index Scan using tickets_pkey on tickets t  (cost=0.00..8.99 rows=1 width=8)"
"              Index Cond: (id_ticket = l.id_ticket)"
"  ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.11 rows=1 width=44)"
"        Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"  SubPlan 3"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 4"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 5"
"    ->  Index Scan using atribconjuninst_pkey on atributoconjuntoinstancia a  (cost=0.00..8.27 rows=1 width=33)"
"          Index Cond: (id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia)"
```

ANEXO 19

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT USANDO JOIN 10 MILLONES DE DATOS.

```
"Merge Join  (cost=966455.01..1074861.89 rows=66514 width=176)"
"  Merge Cond: (cl.id_cliente = t.id_cliente)"
"  ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..101640.91 rows=2240818 width=44)"
"  ->  Materialize  (cost=966454.98..966787.55 rows=66514 width=136)"
"        ->  Sort  (cost=966454.98..966621.26 rows=66514 width=136)"
"              Sort Key: t.id_cliente"
"              ->  Hash Join  (cost=647430.61..956576.75 rows=66514 width=136)"
"                    Hash Cond: (t.id_ticket = l.id_ticket)"
"                    ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=8)"
"                    ->  Hash  (cost=645299.19..645299.19 rows=66514 width=132)"
"                          ->  Nested Loop  (cost=3423.20..645299.19 rows=66514 width=132)"
"                                Join Filter: (p.id_categoria = c.id_categoria)"
"                                ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=1 width=52)"
"                                      Filter: ((nombre)::text = 'FARMACEUTICOS'::text)"
"                                ->  Hash Join  (cost=3423.20..643635.31 rows=133028 width=84)"
"                                      Hash Cond: (l.id_producto = p.id_producto)"
"                                      ->  Hash Join  (cost=28.04..634457.18 rows=135621 width=50)"
"                                            Hash Cond: (l.id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                                            ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=17)"
"                                            ->  Hash  (cost=27.97..27.97 rows=5 width=37)"
"                                                  ->  Seq Scan on atributoconjuntoinstancia aci  (cost=0.00..27.97 rows=5 width=37)"
"                                                        Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO
VITAFARMA","ATRIB.CONJUNTO QUIFATEX GENERICOS"}'::text[]))"
"                                      ->  Hash  (cost=1794.63..1794.63 rows=78763 width=38)"
"                                            ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=38)"
```

ANEXO 20

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT USANDO SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
"Nested Loop  (cost=0.00..280271390.22 rows=678 width=94)"
"  ->  Nested Loop  (cost=0.00..280242730.34 rows=678 width=54)"
"        ->  Nested Loop  (cost=0.00..280236628.40 rows=678 width=50)"
"              Join Filter: (l.id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..280225350.72 rows=149996 width=17)"
"                    Filter: (((SubPlan 5))::text = 'FARMACEUTICOS'::text)"
"                    SubPlan 5"
"                      ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"                            Join Filter: (p.id_categoria = c.id_categoria)"
"                            ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                                  Index Cond: (id_producto = l.id_producto)"
"                            ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"              ->  Materialize  (cost=0.00..28.00 rows=5 width=37)"
"                    ->  Seq Scan on atributoconjuntoinstancia aci  (cost=0.00..27.97 rows=5 width=37)"
"                          Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO VITAFARMA","ATRIB.CONJUNTO
QUIFATEX GENERICOS"}'::text[]))"
"        ->  Index Scan using tickets_pkey on tickets t  (cost=0.00..8.99 rows=1 width=8)"
"              Index Cond: (id_ticket = l.id_ticket)"
"  ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.11 rows=1 width=44)"
"        Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"  SubPlan 3"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 4"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
```

ANEXO 21

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT USANDO ÍNDICE + JOIN 10 MILLONES DE DATOS..

```
"Hash Join  (cost=716678.81..840670.27 rows=66514 width=176)"
"  Hash Cond: (cl.id_cliente = t.id_cliente)"
"  ->  Seq Scan on clientes cl  (cost=0.00..65830.18 rows=2240818 width=44)"
"  ->  Hash  (cost=714547.39..714547.39 rows=66514 width=136)"
"        ->  Hash Join  (cost=405401.25..714547.39 rows=66514 width=136)"
"              Hash Cond: (t.id_ticket = l.id_ticket)"
"              ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=8)"
"              ->  Hash  (cost=403269.82..403269.82 rows=66514 width=132)"
"                    ->  Nested Loop  (cost=3908.92..403269.82 rows=66514 width=132)"
"                          Join Filter: (p.id_categoria = c.id_categoria)"
"                          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=1 width=52)"
"                                Filter: ((nombre)::text = 'FARMACEUTICOS'::text)"
"                          ->  Hash Join  (cost=3908.92..401605.95 rows=133028 width=84)"
"                                Hash Cond: (l.id_producto = p.id_producto)"
"                                ->  Nested Loop  (cost=513.75..392427.82 rows=135621 width=50)"
"                                      ->  Seq Scan on atributoconjuntoinstancia aci  (cost=0.00..27.97 rows=5 width=37)"
"                                            Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO
VITAFARMA","ATRIB.CONJUNTO QUIFATEX GENERICOS"}'::text[]))"
"                                      ->  Bitmap Heap Scan on lineasticket l  (cost=513.75..78208.73 rows=27124 width=17)"
"                                            Recheck Cond: (id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                                            ->  Bitmap Index Scan on atribconjinst_descrip  (cost=0.00..506.97 rows=27124 width=0)"
"                                                  Index Cond: (id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                                ->  Hash  (cost=1794.63..1794.63 rows=78763 width=38)"
"                                      ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=38)"
```

ANEXO 22

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT CON ÍNDICE + SUBQUERY 10 MILLONES DE DATOS.

```
"Nested Loop  (cost=507.00..1695858.13 rows=678 width=61)"
"  ->  Nested Loop  (cost=507.00..1661592.72 rows=678 width=21)"
"        ->  Nested Loop  (cost=507.00..1655490.79 rows=678 width=17)"
"              ->  Seq Scan on atributoconjuntoinstancia a  (cost=0.00..27.97 rows=5 width=4)"
"                    Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO VITAFARMA","ATRIB.CONJUNTO QUIFATEX
GENERICOS"}'::text[]))"
"              ->  Bitmap Heap Scan on lineasticket l  (cost=507.00..331091.20 rows=136 width=17)"
"                    Recheck Cond: (id_atributoconjuntoinstancia = a.id_atributoconjuntoinstancia)"
"                    Filter: (((SubPlan 6))::text = 'FARMACEUTICOS'::text)"
"                    ->  Bitmap Index Scan on atribconjinst_descrip  (cost=0.00..506.97 rows=27124 width=0)"
"                          Index Cond: (id_atributoconjuntoinstancia = a.id_atributoconjuntoinstancia)"
"                    SubPlan 6"
"                      ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"                            Join Filter: (p.id_categoria = c.id_categoria)"
"                            ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                                  Index Cond: (id_producto = l.id_producto)"
"                            ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"        ->  Index Scan using tickets_pkey on tickets t  (cost=0.00..8.99 rows=1 width=8)"
"              Index Cond: (id_ticket = l.id_ticket)"
"  ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.11 rows=1 width=44)"
"        Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"  SubPlan 3"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 4"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 5"
"    ->  Index Scan using atribconjuninst_pkey on atributoconjuntoinstancia a  (cost=0.00..8.27 rows=1 width=33)"
"          Index Cond: (id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia)"
```

ANEXO 23

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
"Nested Loop  (cost=507.00..1690252.61 rows=678 width=94)"
"  ->  Nested Loop  (cost=507.00..1661592.72 rows=678 width=54)"
"        ->  Nested Loop  (cost=507.00..1655490.79 rows=678 width=50)"
"              ->  Seq Scan on atributoconjuntoinstancia aci  (cost=0.00..27.97 rows=5 width=37)"
"                    Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO VITAFARMA","ATRIB.CONJUNTO QUIFATEX
GENERICOS"}'::text[]))"
"              ->  Bitmap Heap Scan on lineasticket l  (cost=507.00..331091.20 rows=136 width=17)"
"                    Recheck Cond: (id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                    Filter: (((SubPlan 5))::text = 'FARMACEUTICOS'::text)"
"                    ->  Bitmap Index Scan on atribconjinst_descrip  (cost=0.00..506.97 rows=27124 width=0)"
"                          Index Cond: (id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                    SubPlan 5"
"                      ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"                            Join Filter: (p.id_categoria = c.id_categoria)"
"                            ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                                  Index Cond: (id_producto = l.id_producto)"
"                            ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"        ->  Index Scan using tickets_pkey on tickets t  (cost=0.00..8.99 rows=1 width=8)"
"              Index Cond: (id_ticket = l.id_ticket)"
"  ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.11 rows=1 width=44)"
"        Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"  SubPlan 3"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 4"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
```

ANEXO 24

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 3 SELECT SIMPLE Y SELECT USANDO JOIN 10 MILLONES DE DATOS.

```
"GroupAggregate  (cost=5237357.26..5601353.60 rows=9706569 width=91)"
"  Filter: ((sum(l.unidades) >= 600::numeric) AND (sum(l.unidades) <= 1000::numeric))"
"  ->  Sort  (cost=5237357.26..5261623.68 rows=9706569 width=91)"
"        Sort Key: l.id_producto, p.nombre, l.precio, c.id_categoria, c.nombre"
"        ->  Hash Join  (cost=667427.30..2120266.20 rows=9706569 width=91)"
"              Hash Cond: (l.id_producto = p.id_producto)"
"              ->  Hash Join  (cost=662487.10..1833955.59 rows=9895739 width=13)"
"                    Hash Cond: (l.id_ticket = t.id_ticket)"
"                    ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=17)"
"                    ->  Hash  (cost=608367.85..608367.85 rows=3298660 width=8)"
"                          ->  Hash Join  (cost=267814.25..608367.85 rows=3298660 width=8)"
"                                Hash Cond: (t.id_ticket = i.id_ingreso)"
"                                ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=4)"
"                                ->  Hash  (cost=213695.00..213695.00 rows=3298660 width=4)"
"                                      ->  Seq Scan on ingresos i  (cost=0.00..213695.00 rows=3298660 width=4)"
"                                            Filter: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"              ->  Hash  (cost=2878.67..2878.67 rows=78763 width=82)"
"                    ->  Hash Join  (cost=1.04..2878.67 rows=78763 width=82)"
"                          Hash Cond: (p.id_categoria = c.id_categoria)"
"                          ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=34)"
"                          ->  Hash  (cost=1.02..1.02 rows=2 width=52)"
"                                ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
```

ANEXO 25

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 3 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 10 MILLONES DE DATOS.

```
"GroupAggregate  (cost=5206833.77..5570830.11 rows=9706569 width=91)"
"  Filter: ((sum(l.unidades) >= 600::numeric) AND (sum(l.unidades) <= 1000::numeric))"
"  ->  Sort  (cost=5206833.77..5231100.19 rows=9706569 width=91)"
"        Sort Key: l.id_producto, p.nombre, l.precio, c.id_categoria, c.nombre"
"        ->  Hash Join  (cost=636903.82..2089742.72 rows=9706569 width=91)"
"              Hash Cond: (l.id_producto = p.id_producto)"
"              ->  Hash Join  (cost=631963.61..1803432.11 rows=9895739 width=13)"
"                    Hash Cond: (l.id_ticket = t.id_ticket)"
"                    ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=17)"
"                    ->  Hash  (cost=577844.36..577844.36 rows=3298660 width=8)"
"                          ->  Hash Join  (cost=237290.76..577844.36 rows=3298660 width=8)"
"                                Hash Cond: (t.id_ticket = i.id_ingreso)"
"                                ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=4)"
"                                ->  Hash  (cost=183171.51..183171.51 rows=3298660 width=4)"
"                                      ->  Bitmap Heap Scan on ingresos i  (cost=69996.61..183171.51 rows=3298660 width=4)"
"                                            Recheck Cond: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"                                            ->  Bitmap Index Scan on ingresos_inx_1  (cost=0.00..69171.95 rows=3298660 width=0)"
"                                                  Index Cond: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"              ->  Hash  (cost=2878.67..2878.67 rows=78763 width=82)"
"                    ->  Hash Join  (cost=1.04..2878.67 rows=78763 width=82)"
"                          Hash Cond: (p.id_categoria = c.id_categoria)"
"                          ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=34)"
"                          ->  Hash  (cost=1.02..1.02 rows=2 width=52)"
"                                ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
```

ANEXO 26

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 3 SELECT USANDO SUBQUERY 10 MILLONES DE DATOS.

```
"GroupAggregate  (cost=7788780.55..85902859.70 rows=2999927 width=13)"
"  Filter: ((sum(l.unidades) >= 600::numeric) AND (sum(l.unidades) <= 1000::numeric))"
"  ->  Sort  (cost=7788780.55..7863778.72 rows=29999268 width=13)"
"        Sort Key: l.id_producto, l.precio"
"        ->  Hash Semi Join  (cost=639870.08..2012373.86 rows=29999268 width=13)"
"              Hash Cond: (l.id_ticket = tickets.id_ticket)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=17)"
"              ->  Hash  (cost=585750.83..585750.83 rows=3298660 width=8)"
"                    ->  Hash Semi Join  (cost=267814.25..585750.83 rows=3298660 width=8)"
"                          Hash Cond: (tickets.id_ticket = ingresos.id_ingreso)"
"                          ->  Seq Scan on tickets  (cost=0.00..154055.00 rows=10000000 width=4)"
"                          ->  Hash  (cost=213695.00..213695.00 rows=3298660 width=4)"
"                                ->  Seq Scan on ingresos  (cost=0.00..213695.00 rows=3298660 width=4)"
"                                      Filter: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"  SubPlan 3"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
```

ANEXO 27

PLAN DE EJECUCIÓN EN POSTGRESQL PRUEBA 3 CONSULTA CON SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
"GroupAggregate  (cost=4306241.22..175432470.30 rows=9706569 width=39)"
"  Filter: ((sum(l.unidades) >= 600::numeric) AND (sum(l.unidades) <= 1000::numeric))"
"  ->  Sort  (cost=4306241.22..4330507.65 rows=9706569 width=39)"
"        Sort Key: p.nombre, l.id_producto, l.precio"
"        ->  Hash Join  (cost=665805.27..2118106.17 rows=9706569 width=39)"
"              Hash Cond: (l.id_producto = p.id_producto)"
"              ->  Hash Join  (cost=662487.10..1833955.59 rows=9895739 width=13)"
"                    Hash Cond: (l.id_ticket = t.id_ticket)"
"                    ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=17)"
"                    ->  Hash  (cost=608367.85..608367.85 rows=3298660 width=8)"
"                          ->  Hash Join  (cost=267814.25..608367.85 rows=3298660 width=8)"
"                                Hash Cond: (t.id_ticket = i.id_ingreso)"
"                                ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=4)"
"                                ->  Hash  (cost=213695.00..213695.00 rows=3298660 width=4)"
"                                      ->  Seq Scan on ingresos i  (cost=0.00..213695.00 rows=3298660 width=4)"
"                                            Filter: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"              ->  Hash  (cost=1794.63..1794.63 rows=78763 width=30)"
"                    ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=30)"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
```

ANEXO 28

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 3 SELECT CON ÍNDICE + SUBQUERY 10 MILLONES DE DATOS.

```
"GroupAggregate  (cost=7758257.06..85872336.21 rows=2999927 width=13)"
"  Filter: ((sum(l.unidades) >= 600::numeric) AND (sum(l.unidades) <= 1000::numeric))"
"  ->  Sort  (cost=7758257.06..7833255.23 rows=29999268 width=13)"
"        Sort Key: l.id_producto, l.precio"
"        ->  Hash Semi Join  (cost=609346.59..1981850.37 rows=29999268 width=13)"
"              Hash Cond: (l.id_ticket = tickets.id_ticket)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=17)"
"              ->  Hash  (cost=555227.34..555227.34 rows=3298660 width=8)"
"                    ->  Hash Semi Join  (cost=237290.76..555227.34 rows=3298660 width=8)"
"                          Hash Cond: (tickets.id_ticket = ingresos.id_ingreso)"
"                          ->  Seq Scan on tickets  (cost=0.00..154055.00 rows=10000000 width=4)"
"                          ->  Hash  (cost=183171.51..183171.51 rows=3298660 width=4)"
"                                ->  Bitmap Heap Scan on ingresos  (cost=69996.61..183171.51 rows=3298660 width=4)"
"                                      Recheck Cond: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"                                      ->  Bitmap Index Scan on ingresos_inx_1  (cost=0.00..69171.95 rows=3298660 width=0)"
"                                            Index Cond: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 3"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
```

ANEXO 29

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 3 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
"GroupAggregate  (cost=4275717.74..175401946.81 rows=9706569 width=39)"
"  Filter: ((sum(l.unidades) >= 600::numeric) AND (sum(l.unidades) <= 1000::numeric))"
"  ->  Sort  (cost=4275717.74..4299984.16 rows=9706569 width=39)"
"        Sort Key: p.nombre, l.id_producto, l.precio"
"        ->  Hash Join  (cost=635281.78..2087582.68 rows=9706569 width=39)"
"              Hash Cond: (l.id_producto = p.id_producto)"
"              ->  Hash Join  (cost=631963.61..1803432.11 rows=9895739 width=13)"
"                    Hash Cond: (l.id_ticket = t.id_ticket)"
"                    ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=17)"
"                    ->  Hash  (cost=577844.36..577844.36 rows=3298660 width=8)"
"                          ->  Hash Join  (cost=237290.76..577844.36 rows=3298660 width=8)"
"                                Hash Cond: (t.id_ticket = i.id_ingreso)"
"                                ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=4)"
"                                ->  Hash  (cost=183171.51..183171.51 rows=3298660 width=4)"
"                                      ->  Bitmap Heap Scan on ingresos i  (cost=69996.61..183171.51 rows=3298660 width=4)"
"                                            Recheck Cond: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"                                            ->  Bitmap Index Scan on ingresos_inx_1  (cost=0.00..69171.95 rows=3298660 width=0)"
"                                                  Index Cond: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"              ->  Hash  (cost=1794.63..1794.63 rows=78763 width=30)"
"                    ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=30)"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
```

ANEXO 30

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 4 SELECT SIMPLE Y SELECT USANDO JOIN 10 MILLONES DE DATOS.

```
"Hash Join  (cost=1969189.37..2396576.77 rows=673041 width=203)"
"  Hash Cond: (t.id_persona = per.id_persona)"
"  ->  Hash Join  (cost=1969188.15..2387321.23 rows=673041 width=89)"
"        Hash Cond: (t.id_cliente = cl.id_cliente)"
"        ->  Hash Join  (cost=1860028.74..2235288.61 rows=673041 width=67)"
"              Hash Cond: (l.id_producto = p.id_producto)"
"              ->  Hash Join  (cost=1856710.57..2206635.15 rows=686158 width=45)"
"                    Hash Cond: ((li.id_ingreso = t.id_ticket) AND (li.id_impuesto = l.id_impuesto))"
"                    ->  Seq Scan on lineasimpuesto li  (cost=0.00..163704.00 rows=10000000 width=14)"
"                    ->  Hash  (cost=1839717.20..1839717.20 rows=686158 width=55)"
"                          ->  Hash Join  (cost=846441.69..1839717.20 rows=686158 width=55)"
"                                Hash Cond: (l.id_ticket = t.id_ticket)"
"                                ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=21)"
"                                ->  Hash  (cost=841795.63..841795.63 rows=228725 width=34)"
"                                      ->  Hash Join  (cost=539314.38..841795.63 rows=228725 width=34)"
"                                            Hash Cond: (i.id_ingreso = t.id_ticket)"
"                                            ->  Seq Scan on ingresos i  (cost=0.00..163695.00 rows=10000000 width=12)"
"                                            ->  Hash  (cost=535114.31..535114.31 rows=228725 width=22)"
"                                                  ->  Hash Join  (cost=217497.06..535114.31 rows=228725 width=22)"
"                                                        Hash Cond: (t.id_ticket = pg.id_ingreso)"
"                                                        ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=12)"
"                                                        ->  Hash  (cost=213521.00..213521.00 rows=228725 width=10)"
"                                                              ->  Seq Scan on pagos pg  (cost=0.00..213521.00 rows=228725 width=10)"
"                                                                    Filter: ((total >= 500::numeric) AND (total <= 800::numeric))"
"              ->  Hash  (cost=1794.63..1794.63 rows=78763 width=30)"
"                    ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=30)"
"        ->  Hash  (cost=65830.18..65830.18 rows=2240818 width=30)"
"              ->  Seq Scan on clientes cl  (cost=0.00..65830.18 rows=2240818 width=30)"
"  ->  Hash  (cost=1.10..1.10 rows=10 width=122)"
"        ->  Seq Scan on persona per  (cost=0.00..1.10 rows=10 width=122)"
```

ANEXO 31

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 4 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 10 MILLONES DE DATOS.

```
"Hash Join  (cost=1827478.02..2254865.43 rows=673041 width=203)"
"  Hash Cond: (t.id_persona = per.id_persona)"
"  ->  Hash Join  (cost=1827476.80..2245609.89 rows=673041 width=89)"
"        Hash Cond: (t.id_cliente = cl.id_cliente)"
"        ->  Hash Join  (cost=1718317.39..2093577.27 rows=673041 width=67)"
"              Hash Cond: (l.id_producto = p.id_producto)"
"              ->  Hash Join  (cost=1714999.23..2064923.81 rows=686158 width=45)"
"                    Hash Cond: ((li.id_ingreso = t.id_ticket) AND (li.id_impuesto = l.id_impuesto))"
"                    ->  Seq Scan on lineasimpuesto li  (cost=0.00..163704.00 rows=10000000 width=14)"
"                    ->  Hash  (cost=1698005.86..1698005.86 rows=686158 width=55)"
"                          ->  Hash Join  (cost=704730.34..1698005.86 rows=686158 width=55)"
"                                Hash Cond: (l.id_ticket = t.id_ticket)"
"                                ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=21)"
"                                ->  Hash  (cost=700084.28..700084.28 rows=228725 width=34)"
"                                      ->  Hash Join  (cost=397603.03..700084.28 rows=228725 width=34)"
"                                            Hash Cond: (i.id_ingreso = t.id_ticket)"
"                                            ->  Seq Scan on ingresos i  (cost=0.00..163695.00 rows=10000000 width=12)"
"                                            ->  Hash  (cost=393402.97..393402.97 rows=228725 width=22)"
"                                                  ->  Hash Join  (cost=75785.72..393402.97 rows=228725 width=22)"
"                                                        Hash Cond: (t.id_ticket = pg.id_ingreso)"
"                                                        ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=12)"
"                                                        ->  Hash  (cost=71809.65..71809.65 rows=228725 width=10)"
"                                                              ->  Bitmap Heap Scan on pagos pg  (cost=4857.78..71809.65 rows=228725 width=10)"
"                                                                    Recheck Cond: ((total >= 500::numeric) AND (total <= 800::numeric))"
"                                                                    ->  Bitmap Index Scan on pagos_total  (cost=0.00..4800.60 rows=228725 width=0)"
"                                                                          Index Cond: ((total >= 500::numeric) AND (total <= 800::numeric))"
"              ->  Hash  (cost=1794.63..1794.63 rows=78763 width=30)"
"                    ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=30)"
"        ->  Hash  (cost=65830.18..65830.18 rows=2240818 width=30)"
"              ->  Seq Scan on clientes cl  (cost=0.00..65830.18 rows=2240818 width=30)"
"  ->  Hash  (cost=1.10..1.10 rows=10 width=122)"
"        ->  Seq Scan on persona per  (cost=0.00..1.10 rows=10 width=122)"
```

ANEXO 32

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 4 SELECT USANDO SUBQUERY 10 MILLONES DE DATOS.

```
"Nested Loop  (cost=0.00..5111256042053.86 rows=3333252 width=41)"
"  ->  Nested Loop  (cost=0.00..3770430394149.86 rows=1111111 width=24)"
"        ->  Index Scan using ingresos_pkey on ingresos i  (cost=0.00..3770420373387.35 rows=1111111 width=12)"
"              Filter: (((SubPlan 6) >= 500::numeric) AND ((SubPlan 7) <= 800::numeric))"
"              SubPlan 6"
"                ->  Seq Scan on pagos pg  (cost=0.00..188521.00 rows=1 width=6)"
"                      Filter: (id_ingreso = i.id_ingreso)"
"              SubPlan 7"
"                ->  Seq Scan on pagos pg  (cost=0.00..188521.00 rows=1 width=6)"
"                      Filter: (id_ingreso = i.id_ingreso)"
"        ->  Index Scan using tickets_pkey on tickets t  (cost=0.00..9.01 rows=1 width=12)"
"              Index Cond: (id_ticket = i.id_ingreso)"
"  ->  Index Scan using lineasticket_pkey on lineasticket l  (cost=0.00..43.58 rows=20 width=21)"
"        Index Cond: (id_ticket = t.id_ticket)"
"  SubPlan 1"
"    ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.51 rows=1 width=26)"
"          Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 2"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 3"
"    ->  Seq Scan on lineasimpuesto li  (cost=0.00..213704.00 rows=1 width=6)"
"          Filter: ((id_ingreso = i.id_ingreso) AND (l.id_impuesto = id_impuesto))"
"  SubPlan 4"
"    ->  Seq Scan on pagos pg  (cost=0.00..188521.00 rows=1 width=6)"
"          Filter: (id_ingreso = i.id_ingreso)"
"  SubPlan 5"
"    ->  Seq Scan on persona per  (cost=0.00..1.13 rows=1 width=118)"
"          Filter: (id_persona = t.id_persona)"
```

ANEXO 33

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 4 SELECT USANDO SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
"Hash Join  (cost=1856710.57..14498881.26 rows=686158 width=45)"
"  Hash Cond: ((li.id_ingreso = t.id_ticket) AND (li.id_impuesto = l.id_impuesto))"
"  ->  Seq Scan on lineasimpuesto li  (cost=0.00..163704.00 rows=10000000 width=14)"
"  ->  Hash  (cost=1839717.20..1839717.20 rows=686158 width=55)"
"        ->  Hash Join  (cost=846441.69..1839717.20 rows=686158 width=55)"
"              Hash Cond: (l.id_ticket = t.id_ticket)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=21)"
"              ->  Hash  (cost=841795.63..841795.63 rows=228725 width=34)"
"                    ->  Hash Join  (cost=539314.38..841795.63 rows=228725 width=34)"
"                          Hash Cond: (i.id_ingreso = t.id_ticket)"
"                          ->  Seq Scan on ingresos i  (cost=0.00..163695.00 rows=10000000 width=12)"
"                          ->  Hash  (cost=535114.31..535114.31 rows=228725 width=22)"
"                                ->  Hash Join  (cost=217497.06..535114.31 rows=228725 width=22)"
"                                      Hash Cond: (t.id_ticket = pg.id_ingreso)"
"                                      ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=12)"
"                                      ->  Hash  (cost=213521.00..213521.00 rows=228725 width=10)"
"                                            ->  Seq Scan on pagos pg  (cost=0.00..213521.00 rows=228725 width=10)"
"                                                  Filter: ((total >= 500::numeric) AND (total <= 800::numeric))"
"  SubPlan 1"
"    ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.51 rows=1 width=26)"
"          Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 2"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 3"
"    ->  Seq Scan on persona per  (cost=0.00..1.13 rows=1 width=118)"
"          Filter: (id_persona = t.id_persona)"
```

ANEXO 34

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 4 SELECT CON ÍNDICE + SUBQUERY 10 MILLONES DE DATOS.

```
"Hash Join  (cost=704730.34..146648699483.96 rows=686158 width=47)"
"  Hash Cond: (l.id_ticket = t.id_ticket)"
"  ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=21)"
"  ->  Hash  (cost=700084.28..700084.28 rows=228725 width=34)"
"        ->  Hash Join  (cost=397603.03..700084.28 rows=228725 width=34)"
"              Hash Cond: (i.id_ingreso = t.id_ticket)"
"              ->  Seq Scan on ingresos i  (cost=0.00..163695.00 rows=10000000 width=12)"
"              ->  Hash  (cost=393402.97..393402.97 rows=228725 width=22)"
"                    ->  Hash Join  (cost=75785.72..393402.97 rows=228725 width=22)"
"                          Hash Cond: (t.id_ticket = pg.id_ingreso)"
"                          ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=12)"
"                          ->  Hash  (cost=71809.65..71809.65 rows=228725 width=10)"
"                                ->  Bitmap Heap Scan on pagos pg  (cost=4857.78..71809.65 rows=228725 width=10)"
"                                      Recheck Cond: ((total >= 500::numeric) AND (total <= 800::numeric))"
"                                      ->  Bitmap Index Scan on pagos_total  (cost=0.00..4800.60 rows=228725 width=0)"
"                                            Index Cond: ((total >= 500::numeric) AND (total <= 800::numeric))"
"  SubPlan 1"
"    ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.51 rows=1 width=26)"
"          Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 2"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 3"
"    ->  Seq Scan on lineasimpuesto li  (cost=0.00..213704.00 rows=1 width=6)"
"          Filter: ((id_ingreso = i.id_ingreso) AND (l.id_impuesto = id_impuesto))"
"  SubPlan 4"
"    ->  Seq Scan on persona per  (cost=0.00..1.13 rows=1 width=118)"
"          Filter: (id_persona = t.id_persona)"
```

ANEXO 35

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 4 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
"Hash Join  (cost=1714999.23..14357169.91 rows=686158 width=45)"
"  Hash Cond: ((li.id_ingreso = t.id_ticket) AND (li.id_impuesto = l.id_impuesto))"
"  ->  Seq Scan on lineasimpuesto li  (cost=0.00..163704.00 rows=10000000 width=14)"
"  ->  Hash  (cost=1698005.86..1698005.86 rows=686158 width=55)"
"        ->  Hash Join  (cost=704730.34..1698005.86 rows=686158 width=55)"
"              Hash Cond: (l.id_ticket = t.id_ticket)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..520575.68 rows=29999268 width=21)"
"              ->  Hash  (cost=700084.28..700084.28 rows=228725 width=34)"
"                    ->  Hash Join  (cost=397603.03..700084.28 rows=228725 width=34)"
"                          Hash Cond: (i.id_ingreso = t.id_ticket)"
"                          ->  Seq Scan on ingresos i  (cost=0.00..163695.00 rows=10000000 width=12)"
"                          ->  Hash  (cost=393402.97..393402.97 rows=228725 width=22)"
"                                ->  Hash Join  (cost=75785.72..393402.97 rows=228725 width=22)"
"                                      Hash Cond: (t.id_ticket = pg.id_ingreso)"
"                                      ->  Seq Scan on tickets t  (cost=0.00..154055.00 rows=10000000 width=12)"
"                                      ->  Hash  (cost=71809.65..71809.65 rows=228725 width=10)"
"                                            ->  Bitmap Heap Scan on pagos pg  (cost=4857.78..71809.65 rows=228725 width=10)"
"                                                  Recheck Cond: ((total >= 500::numeric) AND (total <= 800::numeric))"
"                                                  ->  Bitmap Index Scan on pagos_total  (cost=0.00..4800.60 rows=228725 width=0)"
"                                                        Index Cond: ((total >= 500::numeric) AND (total <= 800::numeric))"
"  SubPlan 1"
"    ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.51 rows=1 width=26)"
"          Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 2"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 3"
"    ->  Seq Scan on persona per  (cost=0.00..1.13 rows=1 width=118)"
"          Filter: (id_persona = t.id_persona)"
```

ANEXO 36

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 1 SELECT SIMPLE Y SELECT USANDO JOIN 20 MILLONES DE DATOS.

```
"Hash Join  (cost=1557544.45..3559499.43 rows=510690 width=86)"
"  Hash Cond: (l.id_atributoconjuntoinstancia = a.id_atributoconjuntoinstancia)"
"  ->  Hash Join  (cost=1557509.57..3549889.11 rows=510690 width=53)"
"        Hash Cond: (l.id_producto = p.id_producto)"
"        ->  Hash Join  (cost=1554191.40..3529472.50 rows=510690 width=27)"
"              Hash Cond: (l.id_ticket = i.id_ingreso)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=21)"
"              ->  Hash  (cost=1551065.39..1551065.39 rows=170241 width=18)"
"                    ->  Hash Join  (cost=901781.25..1551065.39 rows=170241 width=18)"
"                          Hash Cond: (im.id_ingreso = i.id_ingreso)"
"                          ->  Seq Scan on lineasimpuesto im  (cost=0.00..377406.00 rows=19940462 width=10)"
"                                Filter: (monto < 500::numeric)"
"                          ->  Hash  (cost=898979.88..898979.88 rows=170749 width=8)"
"                                ->  Hash Join  (cost=435924.21..898979.88 rows=170749 width=8)"
"                                      Hash Cond: (i.id_ingreso = t.id_ticket)"
"                                      ->  Seq Scan on ingresos i  (cost=0.00..427389.00 rows=1707491 width=4)"
"                                            Filter: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"                                      ->  Hash  (cost=403111.21..403111.21 rows=2000000 width=4)"
"                                            ->  Hash Join  (cost=2.21..403111.21 rows=2000000 width=4)"
"                                                  Hash Cond: (t.id_persona = ps.id_persona)"
"                                                  ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=8)"
"                                                  ->  Hash  (cost=2.20..2.20 rows=1 width=4)"
"                                                        ->  Hash Join  (cost=1.05..2.20 rows=1 width=4)"
"                                                              Hash Cond: (ps.id_rol = r.id_rol)"
"                                                              ->  Seq Scan on persona ps  (cost=0.00..1.10 rows=10 width=8)"
"                                                              ->  Hash  (cost=1.04..1.04 rows=1 width=4)"
"                                                                    ->  Seq Scan on roles r  (cost=0.00..1.04 rows=1 width=4)"
"                                                                          Filter: ((nombre)::text = 'Dependiente'::text)"
"        ->  Hash  (cost=1794.63..1794.63 rows=78763 width=30)"
"              ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=30)"
"  ->  Hash  (cost=21.06..21.06 rows=1106 width=37)"
"        ->  Seq Scan on atributoconjuntoinstancia a  (cost=0.00..21.06 rows=1106 width=37)"
```

ANEXO 37

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 1 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 20 MILLONES DE DATOS.

```
"Hash Join  (cost=1315552.32..3317544.12 rows=511276 width=86)"
"  Hash Cond: (l.id_atributoconjuntoinstancia = a.id_atributoconjuntoinstancia)"
"  ->  Hash Join  (cost=1315517.44..3307922.81 rows=511276 width=53)"
"        Hash Cond: (l.id_producto = p.id_producto)"
"        ->  Hash Join  (cost=1312199.27..3287487.22 rows=511276 width=27)"
"              Hash Cond: (l.id_ticket = i.id_ingreso)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=21)"
"              ->  Hash  (cost=1309069.82..1309069.82 rows=170436 width=18)"
"                    ->  Hash Join  (cost=659782.73..1309069.82 rows=170436 width=18)"
"                          Hash Cond: (im.id_ingreso = i.id_ingreso)"
"                          ->  Seq Scan on lineasimpuesto im  (cost=0.00..377406.00 rows=19940462 width=10)"
"                                Filter: (monto < 500::numeric)"
"                          ->  Hash  (cost=656977.92..656977.92 rows=170945 width=8)"
"                                ->  Hash Join  (cost=217355.47..656977.92 rows=170945 width=8)"
"                                      Hash Cond: (t.id_ticket = i.id_ingreso)"
"                                      ->  Hash Join  (cost=2.21..403111.21 rows=2000000 width=4)"
"                                            Hash Cond: (t.id_persona = ps.id_persona)"
"                                            ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=8)"
"                                            ->  Hash  (cost=2.20..2.20 rows=1 width=4)"
"                                                  ->  Hash Join  (cost=1.05..2.20 rows=1 width=4)"
"                                                        Hash Cond: (ps.id_rol = r.id_rol)"
"                                                        ->  Seq Scan on persona ps  (cost=0.00..1.10 rows=10 width=8)"
"                                                        ->  Hash  (cost=1.04..1.04 rows=1 width=4)"
"                                                              ->  Seq Scan on roles r  (cost=0.00..1.04 rows=1 width=4)"
"                                                                    Filter: ((nombre)::text = 'Dependiente'::text)"
"                                      ->  Hash  (cost=189307.11..189307.11 rows=1709452 width=4)"
"                                            ->  Bitmap Heap Scan on ingresos i  (cost=36276.33..189307.11 rows=1709452 width=4)"
"                                                  Recheck Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"                                                  ->  Bitmap Index Scan on ingresos_inx_1  (cost=0.00..35848.96 rows=1709452 width=0)"
"                                                        Index Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"        ->  Hash  (cost=1794.63..1794.63 rows=78763 width=30)"
"              ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=30)"
"  ->  Hash  (cost=21.06..21.06 rows=1106 width=37)"
"        ->  Seq Scan on atributoconjuntoinstancia a  (cost=0.00..21.06 rows=1106 width=37)"
```

ANEXO 38

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 1 SELECT USANDO SUBQUERY 20 MILLONES DE DATOS..

```
"Hash Semi Join  (cost=2525749.45..83279291.49 rows=4753934 width=21)"
"  Hash Cond: (lineasimpuesto.id_ingreso = public.tickets.id_ticket)"
"  ->  Hash Semi Join  (cost=1103294.27..3090208.70 rows=4753934 width=29)"
"        Hash Cond: (l.id_ticket = lineasimpuesto.id_ingreso)"
"        ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=21)"
"        ->  Hash  (cost=1075363.17..1075363.17 rows=1702408 width=8)"
"              ->  Hash Semi Join  (cost=455402.64..1075363.17 rows=1702408 width=8)"
"                    Hash Cond: (lineasimpuesto.id_ingreso = public.ingresos.id_ingreso)"
"                    ->  Seq Scan on lineasimpuesto  (cost=0.00..377406.00 rows=19940462 width=4)"
"                          Filter: (monto < 500::numeric)"
"                    ->  Hash  (cost=427389.00..427389.00 rows=1707491 width=4)"
"                          ->  Seq Scan on ingresos  (cost=0.00..427389.00 rows=1707491 width=4)"
"                                Filter: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"  ->  Hash  (cost=1389642.18..1389642.18 rows=2000000 width=4)"
"        ->  Hash Semi Join  (cost=1006783.18..1389642.18 rows=2000000 width=4)"
"              Hash Cond: (public.tickets.id_persona = persona.id_persona)"
"              ->  Seq Scan on tickets  (cost=0.00..308109.00 rows=20000000 width=8)"
"              ->  Hash  (cost=1006783.17..1006783.17 rows=1 width=8)"
"                    ->  Nested Loop Semi Join  (cost=455402.64..1006783.17 rows=1 width=8)"
"                          Join Filter: (persona.id_persona = public.tickets.id_persona)"
"                          ->  Nested Loop Semi Join  (cost=0.00..13.59 rows=1 width=4)"
"                                Join Filter: (persona.id_rol = roles.id_rol)"
"                                ->  Index Scan using persona_pkey on persona  (cost=0.00..12.40 rows=10 width=8)"
"                                ->  Materialize  (cost=0.00..1.04 rows=1 width=4)"
"                                      ->  Seq Scan on roles  (cost=0.00..1.04 rows=1 width=4)"
"                                            Filter: ((nombre)::text = 'Dependiente'::text)"
"                          ->  Hash Semi Join  (cost=455402.64..1006769.33 rows=1707491 width=4)"
"                                Hash Cond: (public.tickets.id_ticket = public.ingresos.id_ingreso)"
"                                ->  Seq Scan on tickets  (cost=0.00..308109.00 rows=20000000 width=8)"
"                                ->  Hash  (cost=427389.00..427389.00 rows=1707491 width=4)"
"                                      ->  Seq Scan on ingresos  (cost=0.00..427389.00 rows=1707491 width=4)"
"                                            Filter: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Index Scan using atribconjuninst_pkey on atributoconjuntoinstancia a  (cost=0.00..8.27 rows=1 width=33)"
"          Index Cond: (id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia)"
```

ANEXO 39

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 1 SELECT USANDO SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
"Hash Join  (cost=1796775.75..46128395.00 rows=2553449 width=27)"
"  Hash Cond: (l.id_ticket = i.id_ingreso)"
"  ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=21)"
"  ->  Hash  (cost=1781147.70..1781147.70 rows=851204 width=18)"
"        ->  Hash Join  (cost=1097460.35..1781147.70 rows=851204 width=18)"
"              Hash Cond: (im.id_ingreso = i.id_ingreso)"
"              ->  Seq Scan on lineasimpuesto im  (cost=0.00..377406.00 rows=19940462 width=10)"
"                    Filter: (monto < 500::numeric)"
"              ->  Hash  (cost=1083453.52..1083453.52 rows=853746 width=8)"
"                    ->  Hash Join  (cost=455409.06..1083453.52 rows=853746 width=8)"
"                          Hash Cond: (t.id_persona = ps.id_persona)"
"                          ->  Hash Join  (cost=455402.64..1068506.55 rows=1707491 width=12)"
"                                Hash Cond: (t.id_ticket = i.id_ingreso)"
"                                ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=8)"
"                                ->  Hash  (cost=427389.00..427389.00 rows=1707491 width=4)"
"                                      ->  Seq Scan on ingresos i  (cost=0.00..427389.00 rows=1707491 width=4)"
"                                            Filter: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"                          ->  Hash  (cost=6.36..6.36 rows=5 width=4)"
"                                ->  Seq Scan on persona ps  (cost=0.00..6.36 rows=5 width=4)"
"                                      Filter: (SubPlan 3)"
"                                      SubPlan 3"
"                                        ->  Seq Scan on roles r  (cost=0.00..1.04 rows=1 width=4)"
"                                              Filter: ((ps.id_rol = id_rol) AND ((nombre)::text = 'Dependiente'::text))"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Index Scan using atribconjuninst_pkey on atributoconjuntoinstancia a  (cost=0.00..8.27 rows=1 width=33)"
"          Index Cond: (id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia)"
```

ANEXO 40

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 1 SELECT USANDO ÍNDICE + SUBQUERY 20 MILLONES DE DATOS.

```
"Hash Semi Join  (cost=2049750.51..82803299.54 rows=4753934 width=21)"
"  Hash Cond: (lineasimpuesto.id_ingreso = public.tickets.id_ticket)"
"  ->  Hash Semi Join  (cost=865310.48..2852231.90 rows=4753934 width=29)"
"        Hash Cond: (l.id_ticket = lineasimpuesto.id_ingreso)"
"        ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=21)"
"        ->  Hash  (cost=837347.94..837347.94 rows=1704363 width=8)"
"              ->  Hash Semi Join  (cost=217353.26..837347.94 rows=1704363 width=8)"
"                    Hash Cond: (lineasimpuesto.id_ingreso = public.ingresos.id_ingreso)"
"                    ->  Seq Scan on lineasimpuesto  (cost=0.00..377406.00 rows=19940462 width=4)"
"                          Filter: (monto < 500::numeric)"
"                    ->  Hash  (cost=189307.11..189307.11 rows=1709452 width=4)"
"                          ->  Bitmap Heap Scan on ingresos  (cost=36276.33..189307.11 rows=1709452 width=4)"
"                                Recheck Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"                                ->  Bitmap Index Scan on ingresos_inx_1  (cost=0.00..35848.96 rows=1709452 width=0)"
"                                      Index Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"  ->  Hash  (cost=1151627.03..1151627.03 rows=2000000 width=4)"
"        ->  Hash Semi Join  (cost=768768.03..1151627.03 rows=2000000 width=4)"
"              Hash Cond: (public.tickets.id_persona = persona.id_persona)"
"              ->  Seq Scan on tickets  (cost=0.00..308109.00 rows=20000000 width=8)"
"              ->  Hash  (cost=768768.02..768768.02 rows=1 width=8)"
"                    ->  Nested Loop Semi Join  (cost=217353.26..768768.02 rows=1 width=8)"
"                          Join Filter: (persona.id_persona = public.tickets.id_persona)"
"                          ->  Nested Loop Semi Join  (cost=0.00..13.59 rows=1 width=4)"
"                                Join Filter: (persona.id_rol = roles.id_rol)"
"                                ->  Index Scan using persona_pkey on persona  (cost=0.00..12.40 rows=10 width=8)"
"                                ->  Materialize  (cost=0.00..1.04 rows=1 width=4)"
"                                      ->  Seq Scan on roles  (cost=0.00..1.04 rows=1 width=4)"
"                                            Filter: ((nombre)::text = 'Dependiente'::text)"
"                          ->  Hash Semi Join  (cost=217353.26..768754.18 rows=1709452 width=4)"
"                                Hash Cond: (public.tickets.id_ticket = public.ingresos.id_ingreso)"
"                                ->  Seq Scan on tickets  (cost=0.00..308109.00 rows=20000000 width=8)"
"                                ->  Hash  (cost=189307.11..189307.11 rows=1709452 width=4)"
"                                      ->  Bitmap Heap Scan on ingresos  (cost=36276.33..189307.11 rows=1709452 width=4)"
"                                            Recheck Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"                                            ->  Bitmap Index Scan on ingresos_inx_1  (cost=0.00..35848.96 rows=1709452 width=0)"
"                                                  Index Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Index Scan using atribconjuninst_pkey on atributoconjuntoinstancia a  (cost=0.00..8.27 rows=1 width=33)"
"          Index Cond: (id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia)"
```

ANEXO 41

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 1 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
"Hash Join  (cost=1558819.39..45938995.52 rows=2556381 width=27)"
"  Hash Cond: (l.id_ticket = i.id_ingreso)"
"  ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=21)"
"  ->  Hash  (cost=1543173.11..1543173.11 rows=852182 width=18)"
"        ->  Hash Join  (cost=859471.98..1543173.11 rows=852182 width=18)"
"              Hash Cond: (im.id_ingreso = i.id_ingreso)"
"              ->  Seq Scan on lineasimpuesto im  (cost=0.00..377406.00 rows=19940462 width=10)"
"                    Filter: (monto < 500::numeric)"
"              ->  Hash  (cost=845448.91..845448.91 rows=854726 width=8)"
"                    ->  Hash Join  (cost=217359.68..845448.91 rows=854726 width=8)"
"                          Hash Cond: (t.id_persona = ps.id_persona)"
"                          ->  Hash Join  (cost=217353.26..830484.78 rows=1709452 width=12)"
"                                Hash Cond: (t.id_ticket = i.id_ingreso)"
"                                ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=8)"
"                                ->  Hash  (cost=189307.11..189307.11 rows=1709452 width=4)"
"                                      ->  Bitmap Heap Scan on ingresos i  (cost=36276.33..189307.11 rows=1709452 width=4)"
"                                            Recheck Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"                                            ->  Bitmap Index Scan on ingresos_inx_1  (cost=0.00..35848.96 rows=1709452 width=0)"
"                                                  Index Cond: ((fecha >= '2013-01-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-01-31 00:00:00'::timestamp without time zone))"
"                          ->  Hash  (cost=6.36..6.36 rows=5 width=4)"
"                                ->  Seq Scan on persona ps  (cost=0.00..6.36 rows=5 width=4)"
"                                      Filter: (SubPlan 3)"
"                                      SubPlan 3"
"                                        ->  Seq Scan on roles r  (cost=0.00..1.04 rows=1 width=4)"
"                                              Filter: ((ps.id_rol = id_rol) AND ((nombre)::text = 'Dependiente'::text))"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Index Scan using atribconjuninst_pkey on atributoconjuntoinstancia a  (cost=0.00..8.27 rows=1 width=33)"
"          Index Cond: (id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia)"
```

ANEXO 42

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT SIMPLE 20 MILLONES DE DATOS.

```
"Hash Join  (cost=2114586.80..2704730.27 rows=135617 width=176)"
"  Hash Cond: (t.id_cliente = cl.id_cliente)"
"  ->  Hash Join  (cost=2001051.40..2563828.57 rows=135617 width=136)"
"        Hash Cond: (i.id_ingreso = t.id_ticket)"
"        ->  Seq Scan on ingresos i  (cost=0.00..327389.00 rows=20000000 width=4)"
"        ->  Hash  (cost=1996574.18..1996574.18 rows=135617 width=140)"
"              ->  Merge Join  (cost=1316726.48..1996574.18 rows=135617 width=140)"
"                    Merge Cond: (t.id_ticket = l.id_ticket)"
"                    ->  Index Scan using tickets_pkey on tickets t  (cost=0.00..627474.44 rows=20000000 width=8)"
"                    ->  Materialize  (cost=1316726.46..1317404.54 rows=135617 width=132)"
"                          ->  Sort  (cost=1316726.46..1317065.50 rows=135617 width=132)"
"                                Sort Key: l.id_ticket"
"                                ->  Nested Loop  (cost=3423.20..1286622.66 rows=135617 width=132)"
"                                      Join Filter: (p.id_categoria = c.id_categoria)"
"                                      ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=1 width=52)"
"                                            Filter: ((nombre)::text = 'FARMACEUTICOS'::text)"
"                                      ->  Hash Join  (cost=3423.20..1283231.26 rows=271230 width=84)"
"                                            Hash Cond: (l.id_producto = p.id_producto)"
"                                            ->  Hash Join  (cost=28.04..1268836.53 rows=271230 width=50)"
"                                                  Hash Cond: (l.id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                                                  ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=17)"
"                                                  ->  Hash  (cost=27.97..27.97 rows=5 width=37)"
"                                                        ->  Seq Scan on atributoconjuntoinstancia aci  (cost=0.00..27.97 rows=5 width=37)"
"                                                              Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO
VITAFARMA","ATRIB.CONJUNTO QUIFATEX GENERICOS"}'::text[]))"
"                                            ->  Hash  (cost=1794.63..1794.63 rows=78763 width=38)"
"                                                  ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=38)"
"  ->  Hash  (cost=65830.18..65830.18 rows=2240818 width=44)"
"        ->  Seq Scan on clientes cl  (cost=0.00..65830.18 rows=2240818 width=44)"
```

ANEXO 43

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT USANDO ÍNDICE 20 MILLONES DE DATOS.

```
"Hash Join  (cost=754187.01..1680274.47 rows=135617 width=176)"
"  Hash Cond: (t.id_cliente = cl.id_cliente)"
"  ->  Hash Join  (cost=640651.60..1539372.77 rows=135617 width=136)"
"        Hash Cond: (l.id_ticket = t.id_ticket)"
"        ->  Nested Loop  (cost=4417.60..814630.13 rows=135617 width=132)"
"              Join Filter: (p.id_categoria = c.id_categoria)"
"              ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=1 width=52)"
"                    Filter: ((nombre)::text = 'FARMACEUTICOS'::text)"
"              ->  Hash Join  (cost=4417.60..811238.73 rows=271230 width=84)"
"                    Hash Cond: (l.id_producto = p.id_producto)"
"                    ->  Nested Loop  (cost=1022.44..796844.00 rows=271230 width=50)"
"                          ->  Seq Scan on atributoconjuntoinstancia aci  (cost=0.00..27.97 rows=5 width=37)"
"                                Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO VITAFARMA","ATRIB.CONJUNTO
QUIFATEX GENERICOS"}'::text[]))"
"                          ->  Bitmap Heap Scan on lineasticket l  (cost=1022.44..158820.75 rows=54246 width=17)"
"                                Recheck Cond: (id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                                ->  Bitmap Index Scan on atribconjinst_descrip  (cost=0.00..1008.88 rows=54246 width=0)"
"                                      Index Cond: (id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                    ->  Hash  (cost=1794.63..1794.63 rows=78763 width=38)"
"                          ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=38)"
"        ->  Hash  (cost=308109.00..308109.00 rows=20000000 width=8)"
"              ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=8)"
"  ->  Hash  (cost=65830.18..65830.18 rows=2240818 width=44)"
"        ->  Seq Scan on clientes cl  (cost=0.00..65830.18 rows=2240818 width=44)"
```

ANEXO 44

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT USANDO SUBQUERY 20 MILLONES DE DATOS..

```
"Nested Loop  (cost=0.00..560533744.46 rows=1356 width=61)"
"  ->  Nested Loop  (cost=0.00..560464962.71 rows=1356 width=21)"
"        ->  Nested Loop Semi Join  (cost=0.00..560451034.08 rows=1356 width=17)"
"              Join Filter: (l.id_atributoconjuntoinstancia = a.id_atributoconjuntoinstancia)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..560428507.52 rows=299981 width=17)"
"                    Filter: (((SubPlan 6))::text = 'FARMACEUTICOS'::text)"
"                    SubPlan 6"
"                      ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"                            Join Filter: (p.id_categoria = c.id_categoria)"
"                            ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                                  Index Cond: (id_producto = l.id_producto)"
"                            ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"              ->  Materialize  (cost=0.00..28.00 rows=5 width=4)"
"                    ->  Seq Scan on atributoconjuntoinstancia a  (cost=0.00..27.97 rows=5 width=4)"
"                          Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO VITAFARMA","ATRIB.CONJUNTO
QUIFATEX GENERICOS"}'::text[]))"
"        ->  Index Scan using tickets_pkey on tickets t  (cost=0.00..10.26 rows=1 width=8)"
"              Index Cond: (id_ticket = l.id_ticket)"
"  ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.30 rows=1 width=44)"
"        Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"  SubPlan 3"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 4"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 5"
"    ->  Index Scan using atribconjuninst_pkey on atributoconjuntoinstancia a  (cost=0.00..8.27 rows=1 width=33)"
"          Index Cond: (id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia)"
```

ANEXO 45

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT USANDO JOIN 20 MILLONES DE DATOS.

```
"Hash Join  (cost=753192.61..2152266.90 rows=135615 width=176)"
"  Hash Cond: (t.id_cliente = cl.id_cliente)"
"  ->  Hash Join  (cost=639657.20..2011365.23 rows=135615 width=136)"
"        Hash Cond: (l.id_ticket = t.id_ticket)"
"        ->  Nested Loop  (cost=3423.20..1286622.66 rows=135615 width=132)"
"              Join Filter: (p.id_categoria = c.id_categoria)"
"              ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=1 width=52)"
"                    Filter: ((nombre)::text = 'FARMACEUTICOS'::text)"
"              ->  Hash Join  (cost=3423.20..1283231.26 rows=271230 width=84)"
"                    Hash Cond: (l.id_producto = p.id_producto)"
"                    ->  Hash Join  (cost=28.04..1268836.53 rows=271230 width=50)"
"                          Hash Cond: (l.id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                          ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=17)"
"                          ->  Hash  (cost=27.97..27.97 rows=5 width=37)"
"                                ->  Seq Scan on atributoconjuntoinstancia aci  (cost=0.00..27.97 rows=5 width=37)"
"                                      Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO VITAFARMA","ATRIB.CONJUNTO
QUIFATEX GENERICOS"}'::text[]))"

"                    ->  Hash  (cost=1794.63..1794.63 rows=78763 width=38)"
"                          ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=38)"
"        ->  Hash  (cost=308109.00..308109.00 rows=20000000 width=8)"
"              ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=8)"
"  ->  Hash  (cost=65830.18..65830.18 rows=2240818 width=44)"
"        ->  Seq Scan on clientes cl  (cost=0.00..65830.18 rows=2240818 width=44)"
```

ANEXO 46

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT USANDO SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
"Nested Loop  (cost=0.00..560522533.40 rows=1356 width=94)"
"  ->  Nested Loop  (cost=0.00..560464962.71 rows=1356 width=54)"
"        ->  Nested Loop  (cost=0.00..560451034.08 rows=1356 width=50)"
"              Join Filter: (l.id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..560428507.52 rows=299981 width=17)"
"                    Filter: (((SubPlan 5))::text = 'FARMACEUTICOS'::text)"
"                    SubPlan 5"
"                      ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"                            Join Filter: (p.id_categoria = c.id_categoria)"
"                            ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                                  Index Cond: (id_producto = l.id_producto)"
"                            ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"              ->  Materialize  (cost=0.00..28.00 rows=5 width=37)"
"                    ->  Seq Scan on atributoconjuntoinstancia aci  (cost=0.00..27.97 rows=5 width=37)"
"                          Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO VITAFARMA","ATRIB.CONJUNTO
QUIFATEX GENERICOS"}'::text[]))"
"        ->  Index Scan using tickets_pkey on tickets t  (cost=0.00..10.26 rows=1 width=8)"
"              Index Cond: (id_ticket = l.id_ticket)"
"  ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.30 rows=1 width=44)"
"        Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"  SubPlan 3"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 4"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
```

ANEXO 47

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT USANDO ÍNDICE + JOIN 20 MILLONES DE DATOS.

```
"Hash Join  (cost=754187.01..1680274.36 rows=135615 width=176)"
"  Hash Cond: (t.id_cliente = cl.id_cliente)"
"  ->  Hash Join  (cost=640651.60..1539372.69 rows=135615 width=136)"
"        Hash Cond: (l.id_ticket = t.id_ticket)"
"        ->  Nested Loop  (cost=4417.60..814630.13 rows=135615 width=132)"
"              Join Filter: (p.id_categoria = c.id_categoria)"
"              ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=1 width=52)"
"                    Filter: ((nombre)::text = 'FARMACEUTICOS'::text)"
"              ->  Hash Join  (cost=4417.60..811238.73 rows=271230 width=84)"
"                    Hash Cond: (l.id_producto = p.id_producto)"
"                    ->  Nested Loop  (cost=1022.44..796844.00 rows=271230 width=50)"
"                          ->  Seq Scan on atributoconjuntoinstancia aci  (cost=0.00..27.97 rows=5 width=37)"
"                                Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO VITAFARMA","ATRIB.CONJUNTO
QUIFATEX GENERICOS"}'::text[]))"
"                          ->  Bitmap Heap Scan on lineasticket l  (cost=1022.44..158820.75 rows=54246 width=17)"
"                                Recheck Cond: (id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                                ->  Bitmap Index Scan on atribconjinst_descrip  (cost=0.00..1008.88 rows=54246 width=0)"
"                                      Index Cond: (id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                    ->  Hash  (cost=1794.63..1794.63 rows=78763 width=38)"
"                          ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=38)"
"        ->  Hash  (cost=308109.00..308109.00 rows=20000000 width=8)"
"              ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=8)"
"  ->  Hash  (cost=65830.18..65830.18 rows=2240818 width=44)"
"        ->  Seq Scan on clientes cl  (cost=0.00..65830.18 rows=2240818 width=44)"
```

ANEXO 48

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT USANDO ÍNDICE + SUBQUERY 20 MILLONES DE DATOS.

```
"Nested Loop  (cost=1008.94..3405648.54 rows=1356 width=61)"
"  ->  Nested Loop  (cost=1008.94..3336866.80 rows=1356 width=21)"
"        ->  Nested Loop  (cost=1008.94..3322938.16 rows=1356 width=17)"
"              ->  Seq Scan on atributoconjuntoinstancia a  (cost=0.00..27.97 rows=5 width=4)"
"                    Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO VITAFARMA","ATRIB.CONJUNTO QUIFATEX
GENERICOS"}'::text[]))"
"              ->  Bitmap Heap Scan on lineasticket l  (cost=1008.94..664579.33 rows=271 width=17)"
"                    Recheck Cond: (id_atributoconjuntoinstancia = a.id_atributoconjuntoinstancia)"
"                    Filter: (((SubPlan 6))::text = 'FARMACEUTICOS'::text)"
"                    ->  Bitmap Index Scan on atribconjinst_descrip  (cost=0.00..1008.88 rows=54246 width=0)"
"                          Index Cond: (id_atributoconjuntoinstancia = a.id_atributoconjuntoinstancia)"
"                    SubPlan 6"
"                      ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"                            Join Filter: (p.id_categoria = c.id_categoria)"
"                            ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                                  Index Cond: (id_producto = l.id_producto)"
"                            ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"        ->  Index Scan using tickets_pkey on tickets t  (cost=0.00..10.26 rows=1 width=8)"
"              Index Cond: (id_ticket = l.id_ticket)"
"  ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.30 rows=1 width=44)"
"        Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"  SubPlan 3"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 4"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 5"
"    ->  Index Scan using atribconjuninst_pkey on atributoconjuntoinstancia a  (cost=0.00..8.27 rows=1 width=33)"
"          Index Cond: (id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia)"
```

ANEXO 49

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 2 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
"Nested Loop  (cost=1008.94..3394437.49 rows=1356 width=94)"
"  ->  Nested Loop  (cost=1008.94..3336866.80 rows=1356 width=54)"
"        ->  Nested Loop  (cost=1008.94..3322938.16 rows=1356 width=50)"
"              ->  Seq Scan on atributoconjuntoinstancia aci  (cost=0.00..27.97 rows=5 width=37)"
"                    Filter: ((descripcion)::text = ANY ('{"ATRIB.CONJUNTO QUIFATEX PROP MERZ","ATRIB.CONJUNTO NATURPHARMA","ATRIB.CONJUNTO GRUNENTHAL-GENERICOS","ATRIB.CONJUNTO VITAFARMA","ATRIB.CONJUNTO QUIFATEX
GENERICOS"}'::text[]))"
"              ->  Bitmap Heap Scan on lineasticket l  (cost=1008.94..664579.33 rows=271 width=17)"
"                    Recheck Cond: (id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                    Filter: (((SubPlan 5))::text = 'FARMACEUTICOS'::text)"
"                    ->  Bitmap Index Scan on atribconjinst_descrip  (cost=0.00..1008.88 rows=54246 width=0)"
"                          Index Cond: (id_atributoconjuntoinstancia = aci.id_atributoconjuntoinstancia)"
"                    SubPlan 5"
"                      ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"                            Join Filter: (p.id_categoria = c.id_categoria)"
"                            ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                                  Index Cond: (id_producto = l.id_producto)"
"                            ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"        ->  Index Scan using tickets_pkey on tickets t  (cost=0.00..10.26 rows=1 width=8)"
"              Index Cond: (id_ticket = l.id_ticket)"
"  ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.30 rows=1 width=44)"
"        Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"  SubPlan 3"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 4"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
```

ANEXO 50

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 3 SELECT SIMPLE Y SELECT USANDO JOIN 20 MILLONES DE DATOS.

```
"GroupAggregate  (cost=11192056.92..11943540.09 rows=20039551 width=91)"
"  Filter: ((sum(l.unidades) >= 600::numeric) AND (sum(l.unidades) <= 1000::numeric))"
"  ->  Sort  (cost=11192056.92..11242155.80 rows=20039551 width=91)"
"        Sort Key: l.id_producto, p.nombre, l.precio, c.id_categoria, c.nombre"
"        ->  Hash Join  (cost=1433782.93..4651939.42 rows=20039551 width=91)"
"              Hash Cond: (l.id_producto = p.id_producto)"
"              ->  Hash Join  (cost=1428842.73..4074480.64 rows=20039551 width=13)"
"                    Hash Cond: (l.id_ticket = t.id_ticket)"
"                    ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=17)"
"                    ->  Hash  (cost=1319244.25..1319244.25 rows=6680278 width=8)"
"                          ->  Hash Join  (cost=536987.47..1319244.25 rows=6680278 width=8)"
"                                Hash Cond: (t.id_ticket = i.id_ingreso)"
"                                ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=4)"
"                                ->  Hash  (cost=427389.00..427389.00 rows=6680278 width=4)"
"                                      ->  Seq Scan on ingresos i  (cost=0.00..427389.00 rows=6680278 width=4)"
"                                            Filter: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"              ->  Hash  (cost=2878.67..2878.67 rows=78763 width=82)"
"                    ->  Hash Join  (cost=1.04..2878.67 rows=78763 width=82)"
"                          Hash Cond: (p.id_categoria = c.id_categoria)"
"                          ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=34)"
"                          ->  Hash  (cost=1.02..1.02 rows=2 width=52)"
"                                ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
```

ANEXO 51

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 3 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 20 MILLONES DE DATOS.

```
"GroupAggregate  (cost=11134008.39..11885491.55 rows=20039551 width=91)"
"  Filter: ((sum(l.unidades) >= 600::numeric) AND (sum(l.unidades) <= 1000::numeric))"
"  ->  Sort  (cost=11134008.39..11184107.26 rows=20039551 width=91)"
"        Sort Key: l.id_producto, p.nombre, l.precio, c.id_categoria, c.nombre"
"        ->  Hash Join  (cost=1375734.40..4593890.89 rows=20039551 width=91)"
"              Hash Cond: (l.id_producto = p.id_producto)"
"              ->  Hash Join  (cost=1370794.19..4016432.10 rows=20039551 width=13)"
"                    Hash Cond: (l.id_ticket = t.id_ticket)"
"                    ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=17)"
"                    ->  Hash  (cost=1261195.72..1261195.72 rows=6680278 width=8)"
"                          ->  Hash Join  (cost=478938.94..1261195.72 rows=6680278 width=8)"
"                                Hash Cond: (t.id_ticket = i.id_ingreso)"
"                                ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=4)"
"                                ->  Hash  (cost=369340.46..369340.46 rows=6680278 width=4)"
"                                      ->  Bitmap Heap Scan on ingresos i  (cost=141747.29..369340.46 rows=6680278 width=4)"
"                                            Recheck Cond: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"                                            ->  Bitmap Index Scan on ingresos_inx_1  (cost=0.00..140077.22 rows=6680278 width=0)"
"                                                  Index Cond: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"              ->  Hash  (cost=2878.67..2878.67 rows=78763 width=82)"
"                    ->  Hash Join  (cost=1.04..2878.67 rows=78763 width=82)"
"                          Hash Cond: (p.id_categoria = c.id_categoria)"
"                          ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=34)"
"                          ->  Hash  (cost=1.02..1.02 rows=2 width=52)"
"                                ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
```

ANEXO 52

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 3 SELECT USANDO SUBQUERY 20 MILLONES DE DATOS..

```
"GroupAggregate  (cost=16214808.01..172436769.17 rows=5999616 width=13)"
"  Filter: ((sum(l.unidades) >= 600::numeric) AND (sum(l.unidades) <= 1000::numeric))"
"  ->  Sort  (cost=16214808.01..16364798.41 rows=59996160 width=13)"
"        Sort Key: l.id_producto, l.precio"
"        ->  Hash Semi Join  (cost=1317290.06..4362494.06 rows=59996160 width=13)"
"              Hash Cond: (l.id_ticket = tickets.id_ticket)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=17)"
"              ->  Hash  (cost=1207691.58..1207691.58 rows=6680278 width=8)"
"                    ->  Hash Semi Join  (cost=536987.47..1207691.58 rows=6680278 width=8)"
"                          Hash Cond: (tickets.id_ticket = ingresos.id_ingreso)"
"                          ->  Seq Scan on tickets  (cost=0.00..308109.00 rows=20000000 width=4)"
"                          ->  Hash  (cost=427389.00..427389.00 rows=6680278 width=4)"
"                                ->  Seq Scan on ingresos  (cost=0.00..427389.00 rows=6680278 width=4)"
"                                      Filter: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
"  SubPlan 3"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
```

ANEXO 53

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 3 SELECT USANDO SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
"GroupAggregate  (cost=9272036.89..362568118.64 rows=20039551 width=39)"
"  Filter: ((sum(l.unidades) >= 600::numeric) AND (sum(l.unidades) <= 1000::numeric))"
"  ->  Sort  (cost=9272036.89..9322135.76 rows=20039551 width=39)"
"        Sort Key: p.nombre, l.id_producto, l.precio"
"        ->  Hash Join  (cost=1432160.90..4649779.39 rows=20039551 width=39)"
"              Hash Cond: (l.id_producto = p.id_producto)"
"              ->  Hash Join  (cost=1428842.73..4074480.64 rows=20039551 width=13)"
"                    Hash Cond: (l.id_ticket = t.id_ticket)"
"                    ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=17)"
"                    ->  Hash  (cost=1319244.25..1319244.25 rows=6680278 width=8)"
"                          ->  Hash Join  (cost=536987.47..1319244.25 rows=6680278 width=8)"
"                                Hash Cond: (t.id_ticket = i.id_ingreso)"
"                                ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=4)"
"                                ->  Hash  (cost=427389.00..427389.00 rows=6680278 width=4)"
"                                      ->  Seq Scan on ingresos i  (cost=0.00..427389.00 rows=6680278 width=4)"
"                                            Filter: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"              ->  Hash  (cost=1794.63..1794.63 rows=78763 width=30)"
"                    ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=30)"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
```

ANEXO 54

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 3 SELECT USANDO ÍNDICE + SUBQUERY 20 MILLONES DE DATOS.

```
"GroupAggregate  (cost=16156759.47..172378720.63 rows=5999616 width=13)"
"  Filter: ((sum(l.unidades) >= 600::numeric) AND (sum(l.unidades) <= 1000::numeric))"
"  ->  Sort  (cost=16156759.47..16306749.87 rows=59996160 width=13)"
"        Sort Key: l.id_producto, l.precio"
"        ->  Hash Semi Join  (cost=1259241.52..4304445.52 rows=59996160 width=13)"
"              Hash Cond: (l.id_ticket = tickets.id_ticket)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=17)"
"              ->  Hash  (cost=1149643.05..1149643.05 rows=6680278 width=8)"
"                    ->  Hash Semi Join  (cost=478938.94..1149643.05 rows=6680278 width=8)"
"                          Hash Cond: (tickets.id_ticket = ingresos.id_ingreso)"
"                          ->  Seq Scan on tickets  (cost=0.00..308109.00 rows=20000000 width=4)"
"                          ->  Hash  (cost=369340.46..369340.46 rows=6680278 width=4)"
"                                ->  Bitmap Heap Scan on ingresos  (cost=141747.29..369340.46 rows=6680278 width=4)"
"                                      Recheck Cond: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"                                      ->  Bitmap Index Scan on ingresos_inx_1  (cost=0.00..140077.22 rows=6680278 width=0)"
"                                            Index Cond: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 3"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
```

ANEXO 55

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 3 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
"GroupAggregate  (cost=9213988.35..362510070.11 rows=20039551 width=39)"
"  Filter: ((sum(l.unidades) >= 600::numeric) AND (sum(l.unidades) <= 1000::numeric))"
"  ->  Sort  (cost=9213988.35..9264087.23 rows=20039551 width=39)"
"        Sort Key: p.nombre, l.id_producto, l.precio"
"        ->  Hash Join  (cost=1374112.36..4591730.85 rows=20039551 width=39)"
"              Hash Cond: (l.id_producto = p.id_producto)"
"              ->  Hash Join  (cost=1370794.19..4016432.10 rows=20039551 width=13)"
"                    Hash Cond: (l.id_ticket = t.id_ticket)"
"                    ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=17)"
"                    ->  Hash  (cost=1261195.72..1261195.72 rows=6680278 width=8)"
"                          ->  Hash Join  (cost=478938.94..1261195.72 rows=6680278 width=8)"
"                                Hash Cond: (t.id_ticket = i.id_ingreso)"
"                                ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=4)"
"                                ->  Hash  (cost=369340.46..369340.46 rows=6680278 width=4)"
"                                      ->  Bitmap Heap Scan on ingresos i  (cost=141747.29..369340.46 rows=6680278 width=4)"
"                                            Recheck Cond: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"                                            ->  Bitmap Index Scan on ingresos_inx_1  (cost=0.00..140077.22 rows=6680278 width=0)"
"                                                  Index Cond: ((fecha >= '2013-05-01 00:00:00'::timestamp without time zone) AND (fecha <= '2013-08-31 00:00:00'::timestamp without time zone))"
"              ->  Hash  (cost=1794.63..1794.63 rows=78763 width=30)"
"                    ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=30)"
"  SubPlan 1"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 2"
"    ->  Nested Loop  (cost=0.00..9.32 rows=1 width=48)"
"          Join Filter: (p.id_categoria = c.id_categoria)"
"          ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=4)"
"                Index Cond: (id_producto = l.id_producto)"
"          ->  Seq Scan on categorias c  (cost=0.00..1.02 rows=2 width=52)"
```

ANEXO 56

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 4 SELECT SIMPLE Y SELECT USANDO JOIN 20 MILLONES DE DATOS.

```
"Hash Join  (cost=3821777.67..4655633.17 rows=1311375 width=203)"
"  Hash Cond: (t.id_persona = per.id_persona)"
"  ->  Hash Join  (cost=3821776.45..4637600.54 rows=1311375 width=89)"
"        Hash Cond: (l.id_producto = p.id_producto)"
"        ->  Hash Join  (cost=3818458.28..4578419.09 rows=1311375 width=67)"
"              Hash Cond: (t.id_cliente = cl.id_cliente)"
"              ->  Hash Join  (cost=3709298.87..4407939.62 rows=1311375 width=45)"
"                    Hash Cond: ((li.id_ingreso = t.id_ticket) AND (li.id_impuesto = l.id_impuesto))"
"                    ->  Seq Scan on lineasimpuesto li  (cost=0.00..327406.00 rows=20000000 width=14)"
"                    ->  Hash  (cost=3676821.25..3676821.25 rows=1311375 width=55)"
"                          ->  Hash Join  (cost=1691115.30..3676821.25 rows=1311375 width=55)"
"                                Hash Cond: (l.id_ticket = t.id_ticket)"
"                                ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=21)"
"                                ->  Hash  (cost=1682234.89..1682234.89 rows=437153 width=34)"
"                                      ->  Hash Join  (cost=1077598.36..1682234.89 rows=437153 width=34)"
"                                            Hash Cond: (i.id_ingreso = t.id_ticket)"
"                                            ->  Seq Scan on ingresos i  (cost=0.00..327389.00 rows=20000000 width=12)"
"                                            ->  Hash  (cost=1069571.94..1069571.94 rows=437153 width=22)"
"                                                  ->  Hash Join  (cost=434642.41..1069571.94 rows=437153 width=22)"
"                                                        Hash Cond: (t.id_ticket = pg.id_ingreso)"
"                                                        ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=12)"
"                                                        ->  Hash  (cost=427043.00..427043.00 rows=437153 width=10)"
"                                                              ->  Seq Scan on pagos pg  (cost=0.00..427043.00 rows=437153 width=10)"
"                                                                    Filter: ((total >= 500::numeric) AND (total <= 800::numeric))"
"              ->  Hash  (cost=65830.18..65830.18 rows=2240818 width=30)"
"                    ->  Seq Scan on clientes cl  (cost=0.00..65830.18 rows=2240818 width=30)"
"        ->  Hash  (cost=1794.63..1794.63 rows=78763 width=30)"
"              ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=30)"
"  ->  Hash  (cost=1.10..1.10 rows=10 width=122)"
"        ->  Seq Scan on persona per  (cost=0.00..1.10 rows=10 width=122)"
```

ANEXO 57

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 4 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 20 MILLONES DE DATOS.

```
"Hash Join  (cost=3537614.23..4371469.73 rows=1311375 width=203)"
"  Hash Cond: (t.id_persona = per.id_persona)"
"  ->  Hash Join  (cost=3537613.00..4353437.10 rows=1311375 width=89)"
"        Hash Cond: (l.id_producto = p.id_producto)"
"        ->  Hash Join  (cost=3534294.83..4294255.65 rows=1311375 width=67)"
"              Hash Cond: (t.id_cliente = cl.id_cliente)"
"              ->  Hash Join  (cost=3425135.43..4123776.18 rows=1311375 width=45)"
"                    Hash Cond: ((li.id_ingreso = t.id_ticket) AND (li.id_impuesto = l.id_impuesto))"
"                    ->  Seq Scan on lineasimpuesto li  (cost=0.00..327406.00 rows=20000000 width=14)"
"                    ->  Hash  (cost=3392657.80..3392657.80 rows=1311375 width=55)"
"                          ->  Hash Join  (cost=1406951.85..3392657.80 rows=1311375 width=55)"
"                                Hash Cond: (l.id_ticket = t.id_ticket)"
"                                ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=21)"
"                                ->  Hash  (cost=1398071.44..1398071.44 rows=437153 width=34)"
"                                      ->  Hash Join  (cost=793434.91..1398071.44 rows=437153 width=34)"
"                                            Hash Cond: (i.id_ingreso = t.id_ticket)"
"                                            ->  Seq Scan on ingresos i  (cost=0.00..327389.00 rows=20000000 width=12)"
"                                            ->  Hash  (cost=785408.50..785408.50 rows=437153 width=22)"
"                                                  ->  Hash Join  (cost=150478.97..785408.50 rows=437153 width=22)"
"                                                        Hash Cond: (t.id_ticket = pg.id_ingreso)"
"                                                        ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=12)"
"                                                        ->  Hash  (cost=142879.56..142879.56 rows=437153 width=10)"
"                                                              ->  Bitmap Heap Scan on pagos pg  (cost=9279.26..142879.56 rows=437153 width=10)"
"                                                                    Recheck Cond: ((total >= 500::numeric) AND (total <= 800::numeric))"
"                                                                    ->  Bitmap Index Scan on pagos_total  (cost=0.00..9169.97 rows=437153 width=0)"
"                                                                          Index Cond: ((total >= 500::numeric) AND (total <= 800::numeric))"
"              ->  Hash  (cost=65830.18..65830.18 rows=2240818 width=30)"
"                    ->  Seq Scan on clientes cl  (cost=0.00..65830.18 rows=2240818 width=30)"
"        ->  Hash  (cost=1794.63..1794.63 rows=78763 width=30)"
"              ->  Seq Scan on productos p  (cost=0.00..1794.63 rows=78763 width=30)"
"  ->  Hash  (cost=1.10..1.10 rows=10 width=122)"
"        ->  Seq Scan on persona per  (cost=0.00..1.10 rows=10 width=122)"
```

ANEXO 58

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 4 SELECT USANDO SUBQUERY 20 MILLONES DE DATOS..

```
"Hash Join  (cost=1691115.30..560516712829.82 rows=1311375 width=47)"
"  Hash Cond: (l.id_ticket = t.id_ticket)"
"  ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=21)"
"  ->  Hash  (cost=1682234.89..1682234.89 rows=437153 width=34)"
"        ->  Hash Join  (cost=1077598.36..1682234.89 rows=437153 width=34)"
"              Hash Cond: (i.id_ingreso = t.id_ticket)"
"              ->  Seq Scan on ingresos i  (cost=0.00..327389.00 rows=20000000 width=12)"
"              ->  Hash  (cost=1069571.94..1069571.94 rows=437153 width=22)"
"                    ->  Hash Join  (cost=434642.41..1069571.94 rows=437153 width=22)"
"                          Hash Cond: (t.id_ticket = pg.id_ingreso)"
"                          ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=12)"
"                          ->  Hash  (cost=427043.00..427043.00 rows=437153 width=10)"
"                                ->  Seq Scan on pagos pg  (cost=0.00..427043.00 rows=437153 width=10)"
"                                      Filter: ((total >= 500::numeric) AND (total <= 800::numeric))"
"  SubPlan 1"
"    ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.51 rows=1 width=26)"
"          Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 2"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 3"
"    ->  Seq Scan on lineasimpuesto li  (cost=0.00..427406.00 rows=1 width=6)"
"          Filter: ((id_ingreso = i.id_ingreso) AND (l.id_impuesto = id_impuesto))"
"  SubPlan 4"
"    ->  Seq Scan on persona per  (cost=0.00..1.13 rows=1 width=118)"
"          Filter: (id_persona = t.id_persona)"
```

ANEXO 59

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 4 SELECT USANDO SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
"Hash Join  (cost=3709298.87..27900698.20 rows=1311375 width=45)"
"  Hash Cond: ((li.id_ingreso = t.id_ticket) AND (li.id_impuesto = l.id_impuesto))"
"  ->  Seq Scan on lineasimpuesto li  (cost=0.00..327406.00 rows=20000000 width=14)"
"  ->  Hash  (cost=3676821.25..3676821.25 rows=1311375 width=55)"
"        ->  Hash Join  (cost=1691115.30..3676821.25 rows=1311375 width=55)"
"              Hash Cond: (l.id_ticket = t.id_ticket)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=21)"
"              ->  Hash  (cost=1682234.89..1682234.89 rows=437153 width=34)"
"                    ->  Hash Join  (cost=1077598.36..1682234.89 rows=437153 width=34)"
"                          Hash Cond: (i.id_ingreso = t.id_ticket)"
"                          ->  Seq Scan on ingresos i  (cost=0.00..327389.00 rows=20000000 width=12)"
"                          ->  Hash  (cost=1069571.94..1069571.94 rows=437153 width=22)"
"                                ->  Hash Join  (cost=434642.41..1069571.94 rows=437153 width=22)"
"                                      Hash Cond: (t.id_ticket = pg.id_ingreso)"
"                                      ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=12)"
"                                      ->  Hash  (cost=427043.00..427043.00 rows=437153 width=10)"
"                                            ->  Seq Scan on pagos pg  (cost=0.00..427043.00 rows=437153 width=10)"
"                                                  Filter: ((total >= 500::numeric) AND (total <= 800::numeric))"
"  SubPlan 1"
"    ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.51 rows=1 width=26)"
"          Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 2"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 3"
"    ->  Seq Scan on persona per  (cost=0.00..1.13 rows=1 width=118)"
"          Filter: (id_persona = t.id_persona)"
```

ANEXO 60

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 4 SELECT USANDO ÍNDICE + SUBQUERY 20 MILLONES DE DATOS.

```
"Hash Join  (cost=1406951.85..560516428666.38 rows=1311375 width=47)"
"  Hash Cond: (l.id_ticket = t.id_ticket)"
"  ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=21)"
"  ->  Hash  (cost=1398071.44..1398071.44 rows=437153 width=34)"
"        ->  Hash Join  (cost=793434.91..1398071.44 rows=437153 width=34)"
"              Hash Cond: (i.id_ingreso = t.id_ticket)"
"              ->  Seq Scan on ingresos i  (cost=0.00..327389.00 rows=20000000 width=12)"
"              ->  Hash  (cost=785408.50..785408.50 rows=437153 width=22)"
"                    ->  Hash Join  (cost=150478.97..785408.50 rows=437153 width=22)"
"                          Hash Cond: (t.id_ticket = pg.id_ingreso)"
"                          ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=12)"
"                          ->  Hash  (cost=142879.56..142879.56 rows=437153 width=10)"
"                                ->  Bitmap Heap Scan on pagos pg  (cost=9279.26..142879.56 rows=437153 width=10)"
"                                      Recheck Cond: ((total >= 500::numeric) AND (total <= 800::numeric))"
"                                      ->  Bitmap Index Scan on pagos_total  (cost=0.00..9169.97 rows=437153 width=0)"
"                                            Index Cond: ((total >= 500::numeric) AND (total <= 800::numeric))"
"  SubPlan 1"
"    ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.51 rows=1 width=26)"
"          Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 2"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 3"
"    ->  Seq Scan on lineasimpuesto li  (cost=0.00..427406.00 rows=1 width=6)"
"          Filter: ((id_ingreso = i.id_ingreso) AND (l.id_impuesto = id_impuesto))"
"  SubPlan 4"
"    ->  Seq Scan on persona per  (cost=0.00..1.13 rows=1 width=118)"
"          Filter: (id_persona = t.id_persona)"
```

ANEXO 61

PLAN DE EJECUCIÓN EN POSTGRESQL CONSULTA 4 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
"Hash Join  (cost=3425135.43..27616534.75 rows=1311375 width=45)"
"  Hash Cond: ((li.id_ingreso = t.id_ticket) AND (li.id_impuesto = l.id_impuesto))"
"  ->  Seq Scan on lineasimpuesto li  (cost=0.00..327406.00 rows=20000000 width=14)"
"  ->  Hash  (cost=3392657.80..3392657.80 rows=1311375 width=55)"
"        ->  Hash Join  (cost=1406951.85..3392657.80 rows=1311375 width=55)"
"              Hash Cond: (l.id_ticket = t.id_ticket)"
"              ->  Seq Scan on lineasticket l  (cost=0.00..1041110.60 rows=59996160 width=21)"
"              ->  Hash  (cost=1398071.44..1398071.44 rows=437153 width=34)"
"                    ->  Hash Join  (cost=793434.91..1398071.44 rows=437153 width=34)"
"                          Hash Cond: (i.id_ingreso = t.id_ticket)"
"                          ->  Seq Scan on ingresos i  (cost=0.00..327389.00 rows=20000000 width=12)"
"                          ->  Hash  (cost=785408.50..785408.50 rows=437153 width=22)"
"                                ->  Hash Join  (cost=150478.97..785408.50 rows=437153 width=22)"
"                                      Hash Cond: (t.id_ticket = pg.id_ingreso)"
"                                      ->  Seq Scan on tickets t  (cost=0.00..308109.00 rows=20000000 width=12)"
"                                      ->  Hash  (cost=142879.56..142879.56 rows=437153 width=10)"
"                                            ->  Bitmap Heap Scan on pagos pg  (cost=9279.26..142879.56 rows=437153 width=10)"
"                                                  Recheck Cond: ((total >= 500::numeric) AND (total <= 800::numeric))"
"                                                  ->  Bitmap Index Scan on pagos_total  (cost=0.00..9169.97 rows=437153 width=0)"
"                                                        Index Cond: ((total >= 500::numeric) AND (total <= 800::numeric))"
"  SubPlan 1"
"    ->  Index Scan using clientes_pkey on clientes cl  (cost=0.00..8.51 rows=1 width=26)"
"          Index Cond: (id_cliente = t.id_cliente)"
"  SubPlan 2"
"    ->  Index Scan using productos_pkey on productos p  (cost=0.00..8.28 rows=1 width=26)"
"          Index Cond: (id_producto = l.id_producto)"
"  SubPlan 3"
"    ->  Seq Scan on persona per  (cost=0.00..1.13 rows=1 width=118)"
"          Filter: (id_persona = t.id_persona)"
```

ANEXO 62

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 1 SELECT SIMPLE Y SELECT USANDO JOIN 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	943224	0	3,594242	272,6965
|--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	943224	0	2,234366	269,1022
|--Bitmap(HASH:([p].[id_producto]), DEFINE:([Opt_Bitmap1026]))	Bitmap	Bitmap Create	78763	0	0,1585623	0,4493123
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,1585623	0,4493123
| |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,02169907	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	708739	0	2,066878	266,4185
|--Hash Match(Inner Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open10].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]=[open10].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Inner Join	708739	0	1,966329	264,3517
|--Bitmap(HASH:([a].[id_atributoconjuntoinstancia]), DEFINE:([Opt_Bitmap1024]))	Bitmap	Bitmap Create	1106	0	0,03032635	0,03897994
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,03032635	0,03897994
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,0003434	0,008653585
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	708739	0	1,321108	262,3463
|--Hash Match(Inner Join, HASH:([t].[id_ticket])=([l].[id_ticket]))	Hash Match	Inner Join	708739	0	9,496265	261,0252
|--Bitmap(HASH:([t].[id_ticket]), DEFINE:([Opt_Bitmap1023]))	Bitmap	Bitmap Create	3333334	0	2,025375	44,00307
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	3333334	0	2,025375	44,00307
| |--Hash Match(Inner Join, HASH:([ps].[id_persona])=([t].[id_persona]))	Hash Match	Inner Join	3333334	0	16,25242	41,97769
| |--Bitmap(HASH:([ps].[id_persona]), DEFINE:([Bitmap1032]))	Bitmap	Bitmap Create	3	0	0,02850625	0,03655184
| | |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	3	0	0,02850625	0,03655184
| | |--Nested Loops(Inner Join, OUTER REFERENCES:([ps].[id_rol]))	Nested Loops	Inner Join	3	0	0,000018	0,0080456
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[persona].[persona_id_persona] AS [ps]))	Clustered Index Scan	Clustered Index Scan	10	0,003125	0,000168	0,003293
| | |--Clustered Index Seek(OBJECT:([open10].[dbo].[roles].[roles_id_rol] AS [r]), SEEK:([r].[id_rol]=[open10].[dbo].[persona].[id_rol] as [ps].[id_rol]), WHERE:([open10].[dbo].[roles].[nombre] as [r].[nombre]='Dependiente') ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,004706
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1032],[open10].[dbo].[tickets].[id_persona] as [t].[id_persona],N'[IN ROW]'))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
|--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	708739	0	3,353081	207,5259
|--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1022]))	Bitmap	Bitmap Create	813053	0	0,5155694	28,40355
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	813053	0	0,5155694	28,40355
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_id_ingreso] AS [i]), WHERE:([open10].[dbo].[ingresos].[fecha] as [i].[fecha]>='2013-01-01 00:00:00.000' AND [open10].[dbo].[ingresos].[fecha] as [i].[fecha]<='2013-01-31 00:00:00.000'))	Clustered Index Scan	Clustered Index Scan	813053	22,93794	2,750039	25,68798
|--Hash Match(Inner Join, HASH:([l].[id_ticket])=([im].[id_ingreso]))	Hash Match	Inner Join	708739	0	15,19446	175,7693
|--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1033]))	Bitmap	Bitmap Create	610906	0	0,01527266	122,1632
| |--Compute Scalar(DEFINE:([Expr1016]=CONVERT_IMPLICIT(numeric(10,0),[open10].[dbo].[lineasticket].[unidades] as [l].[unidades]*[open10].[dbo].[lineasticket].[precio] as [l].[precio],0)*(0.12)))	Compute Scalar	Compute Scalar	610906	0	0,01527266	122,1632
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	610906	0	0,858149	122,1479
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1022],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1023],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1024],[open10].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia],N'[IN ROW]') AND PROBE([Opt_Bitmap1026],[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto],N'[IN ROW]'))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([im].[id_ingreso]))	Parallelism	Repartition Streams	9986423	0	7,905915	38,41167
|--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [im]), WHERE:([open10].[dbo].[lineasimpuesto].[monto] as [im].[monto]<(500) AND PROBE([Bitmap1033],[open10].[dbo].[lineasimpuesto].[id_ingreso] as [im].[id_ingreso],N'[IN ROW]'))	Clustered Index Scan	Clustered Index Scan	9986423	26,55572	2,750039	29,30576

ANEXO 63

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 1 SELECT USANDO ÍNDICE 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	980490	0	3,735121	200,5128
|--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	980490	0	2,291619	196,7776
|--Bitmap(HASH:([p].[id_producto]), DEFINE:([Opt_Bitmap1026]))	Bitmap	Bitmap Create	78763	0	0,1585623	0,4493123
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,1585623	0,4493123
| |--Index Scan(OBJECT:([open11].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,02169907	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	733355	0	2,137673	194,0367
|--Hash Match(Inner Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open11].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]=[open11].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Inner Join	733355	0	2,035303	191,899
|--Bitmap(HASH:([a].[id_atributoconjuntoinstancia]), DEFINE:([Opt_Bitmap1024]))	Bitmap	Bitmap Create	1106	0	0,03032635	0,03897994
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,03032635	0,03897994
| |--Clustered Index Scan(OBJECT:([open11].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,0003434	0,008653585
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	733355	0	1,366001	189,8248
|--Hash Match(Inner Join, HASH:([t].[id_ticket])=([l].[id_ticket]))	Hash Match	Inner Join	733355	0	9,543274	188,4588
|--Bitmap(HASH:([t].[id_ticket]), DEFINE:([Opt_Bitmap1023]))	Bitmap	Bitmap Create	3333334	0	2,025375	11,5685
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	3333334	0	2,025375	11,5685
| |--Nested Loops(Inner Join, OUTER REFERENCES:([ps].[id_persona]) OPTIMIZED)	Nested Loops	Inner Join	3333334	0	3,483334	9,543129
| |--Nested Loops(Inner Join, OUTER REFERENCES:([ps].[id_rol]))	Nested Loops	Inner Join	3	0	0,000010	0,03654058
| | |--Parallelism(Distribute Streams, RoundRobin Partitioning)	Parallelism	Distribute Streams	10	0	0,02852632	0,03181932
| | | |--Clustered Index Scan(OBJECT:([open11].[dbo].[persona].[persona_id_persona] AS [ps]))	Clustered Index Scan	Clustered Index Scan	10	0,003125	1,68E-04	0,003293
| | |--Clustered Index Seek(OBJECT:([open11].[dbo].[roles].[roles_id_rol] AS [r]), SEEK:([r].[id_rol]=[open11].[dbo].[persona].[id_rol] as [ps].[id_rol]), WHERE:([open11].[dbo].[roles].[nombre] as [r].[nombre]='Dependiente') ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,004706
| |--Index Seek(OBJECT:([open11].[dbo].[tickets].[Ix_persona_tickets>] AS [t]), SEEK:([t].[id_persona]=[open11].[dbo].[persona].[id_persona] as [ps].[id_persona]) ORDERED FORWARD)	Index Seek	Index Seek	1000000	1,009792	1,100157	6,023255
|--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	733355	0	3,484423	167,347
|--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1022]))	Bitmap	Bitmap Create	847651	0	0,536296	1,937009
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	847651	0	0,536296	1,937009
| |--Index Seek(OBJECT:([open11].[dbo].[ingresos].[Ix_ingreso_fecha] AS [i]), SEEK:([i].[fecha] >= '2013-01-01 00:00:00.000' AND [i].[fecha] <= '2013-01-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	847651	1,167569	0,2331433	1,400713
|--Hash Match(Inner Join, HASH:([l].[id_ticket])=([im].[id_ingreso]))	Hash Match	Inner Join	733355	0	15,33948	161,9255
|--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1032]))	Bitmap	Bitmap Create	633976	0	0,0158494	122,1936
| |--Compute Scalar(DEFINE:([Expr1016]=CONVERT_IMPLICIT(numeric(10,0),[open11].[dbo].[lineasticket].[unidades] as [l].[unidades]*[open11].[dbo].[lineasticket].[precio] as [l].[precio],0)*(0.12)))	Compute Scalar	Compute Scalar	633976	0	0,0158494	122,1936
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	633976	0	0,8894792	122,1777
| |--Clustered Index Scan(OBJECT:([open11].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1022],[open11].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1023],[open11].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1024],[open11].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia],N'[IN ROW]') AND PROBE([Opt_Bitmap1026],[open11].[dbo].[lineasticket].[id_producto] as [l].[id_producto],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9387	8,249838	110,1885
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([im].[id_ingreso]))	Parallelism	Repartition Streams	9990688	0	7,909279	24,39248
|--Index Seek(OBJECT:([open11].[dbo].[lineasimpuesto].[limpuesto_monto] AS [im]), SEEK:([im].[monto] < (500)), WHERE:(PROBE([Bitmap1032],[open11].[dbo].[lineasimpuesto].[id_ingreso] as [im].[id_ingreso],N'[IN ROW]')) ORDERED FORWARD)	Index Seek	Index Seek	9990688	13,73572	2,747478	16,4832

ANEXO 64

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 1 SELECT USANDO SUBQUERY 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtreeCost
|--Compute Scalar(DEFINE:([Expr1031]=[open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]))	Compute Scalar	Compute Scalar	2437228	0	0,243723	390,7491
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	2437228	0	5,612993	390,5054
|--Hash Match(Right Outer Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open10].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]=[open10].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Right Outer Join	2437228	0	6,795257	384,8924
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,030326	0,03897994
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,000343	0,008653585
|--Compute Scalar(DEFINE:([Expr1027]=[open10].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	2437228	0	0,060931	378,0582
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	2437228	0	6,575656	377,9973
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	2437228	0	5,345473	371,4216
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	676316	0	0,857357	365,6268
|--Hash Match(Inner Join, HASH:([open10].[dbo].[persona].[id_persona])=([open10].[dbo].[tickets].[id_persona]))	Hash Match	Inner Join	676316	0	3,885147	364,7694
|--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	3	0	0,028505	40,18967
| |--Nested Loops(Left Semi Join, OUTER REFERENCES:([open10].[dbo].[persona].[id_persona]))	Nested Loops	Left Semi Join	3	0	0,000014	40,16117
| |--Parallelism(Gather Streams)	Parallelism	Gather Streams	3	0	0,028502	0,06504245
| | |--Nested Loops(Inner Join, OUTER REFERENCES:([open10].[dbo].[persona].[id_rol]))	Nested Loops	Inner Join	3	0	0,000010	0,03654058
| | |--Parallelism(Distribute Streams, RoundRobin Partitioning)	Parallelism	Distribute Streams	10	0	0,028526	0,03181932
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[persona].[persona_id_persona]))	Clustered Index Scan	Clustered Index Scan	10	0,003125	0,000168	0,003293
| |--Clustered Index Seek(OBJECT:([open10].[dbo].[roles].[roles_id_rol]), SEEK:([open10].[dbo].[roles].[id_rol]=[open10].[dbo].[persona].[id_rol]), WHERE:([open10].[dbo].[roles].[nombre]='Dependiente') ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,004706
| |--Merge Join(Left Semi Join, MERGE:([open10].[dbo].[tickets].[id_ticket])=([open10].[dbo].[ingresos].[id_ingreso]), RESIDUAL:([open10].[dbo].[tickets].[id_ticket]=[open10].[dbo].[ingresos].[id_ingreso]))	Merge Join	Left Semi Join	13620	0	3,821141	40,09611
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket]), WHERE:([open10].[dbo].[persona].[id_persona]=[open10].[dbo].[tickets].[id_persona]) ORDERED FORWARD)	Clustered Index Scan	Clustered Index Scan	167518	22,93868	11,000160	15,16529
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_id_ingreso]), WHERE:([open10].[dbo].[ingresos].[fecha]>='2013-01-01 00:00:00.000' AND [open10].[dbo].[ingresos].[fecha]<='2013-01-31 00:00:00.000') ORDERED FORWARD)	Clustered Index Scan	Clustered Index Scan	136201	22,93794	11,000160	15,16487
|--Hash Match(Inner Join, HASH:([open10].[dbo].[tickets].[id_ticket])=([l].[id_ticket]))	Hash Match	Inner Join	2437228	0	38,558470	320,6946
|--Bitmap(HASH:([open10].[dbo].[tickets].[id_ticket]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	812429	0	14,057870	130,4268
| |--Hash Match(Inner Join, HASH:([open10].[dbo].[lineasimpuesto].[id_ingreso])=([open10].[dbo].[tickets].[id_ticket]))	Hash Match	Inner Join	812429	0	14,057870	130,4268
| |--Bitmap(HASH:([open10].[dbo].[lineasimpuesto].[id_ingreso]), DEFINE:([Bitmap1033]))	Bitmap	Bitmap Create	812429	0	3,699206	82,76357
| | |--Hash Match(Aggregate, HASH:([open10].[dbo].[lineasimpuesto].[id_ingreso]))	Hash Match	Aggregate	812429	0	3,699206	82,76357
| | |--Hash Match(Inner Join, HASH:([open10].[dbo].[ingresos].[id_ingreso])=([open10].[dbo].[lineasimpuesto].[id_ingreso]))	Hash Match	Inner Join	943247	0	14,144060	79,06436
| | |--Bitmap(HASH:([open10].[dbo].[ingresos].[id_ingreso]), DEFINE:([Bitmap1032]))	Bitmap	Bitmap Create	813053	0	0,515569	28,40355
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open10].[dbo].[ingresos].[id_ingreso]))	Parallelism	Repartition Streams	813053	0	0,515569	28,40355
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_id_ingreso]), WHERE:([open10].[dbo].[ingresos].[fecha]>='2013-01-01 00:00:00.000' AND [open10].[dbo].[ingresos].[fecha]<='2013-01-31 00:00:00.000'))	Clustered Index Scan	Clustered Index Scan	813053	22,93794	2,750039	25,68798
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open10].[dbo].[lineasimpuesto].[id_ingreso]))	Parallelism	Repartition Streams	9986423	0	6,010992	36,51675
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos]), WHERE:([open10].[dbo].[lineasimpuesto].[monto]<(500) AND PROBE([Bitmap1032],[open10].[dbo].[lineasimpuesto].[id_ingreso],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	9986423	26,55572	2,750039	29,30576
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open10].[dbo].[tickets].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	7,916625	33,60535
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket]), WHERE:(PROBE([Bitmap1033],[open10].[dbo].[tickets].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
|--Compute Scalar(DEFINE:([Expr1023]=CONVERT_IMPLICIT(numeric(10,0),[open10].[dbo].[lineasticket].[unidades] as [l].[unidades]*[open10].[dbo].[lineasticket].[precio] as [l].[precio],0)*(0.12)))	Compute Scalar	Compute Scalar	29999270	0	0,749982	151,7094
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	29999270	0	40,769380	150,9594
|--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Bitmap1034],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19

ANEXO 65

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 1 SELECT USANDO SUBQUERY + JOIN 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1020]=[open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]))	Compute Scalar	Compute Scalar	943224	0	0,094322	270,5539
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	943224	0	3,412888	270,4596
|--Hash Match(Right Outer Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open10].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]=[open10].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Right Outer Join	943224	0	2,606544	267,0467
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,030326	0,03897994
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,000343	0,008653585
|--Compute Scalar(DEFINE:([Expr1016]=[open10].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	943224	0	0,023581	264,4012
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	943224	0	2,741272	264,3776
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	943224	0	2,492300	261,6363
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	943224	0	1,748764	258,6947
|--Hash Match(Inner Join, HASH:([t].[id_ticket])=([l].[id_ticket]))	Hash Match	Inner Join	943224	0	9,944073	256,946
|--Bitmap(HASH:([t].[id_ticket]), DEFINE:([Opt_Bitmap1023]))	Bitmap	Bitmap Create	3333334	0	0,000000	42,0062
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	3333334	0	0,028502	42,0062
| |--Hash Match(Inner Join, HASH:([ps].[id_persona])=([t].[id_persona]))	Hash Match	Inner Join	3333334	0	16,252420	41,97769
| |--Bitmap(HASH:([ps].[id_persona]), DEFINE:([Bitmap1027]))	Bitmap	Bitmap Create	3	0	0,028506	0,03655184
| | |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	3	0	0,028506	0,03655184
| | |--Nested Loops(Inner Join, OUTER REFERENCES:([ps].[id_rol]))	Nested Loops	Inner Join	3	0	0,000042	0,0080456
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[persona].[persona_id_persona] AS [ps]))	Clustered Index Scan	Clustered Index Scan	10	0,003125	0,000168	0,003293
| | |--Clustered Index Seek(OBJECT:([open10].[dbo].[roles].[roles_id_rol] AS [r]), SEEK:([r].[id_rol]=[open10].[dbo].[persona].[id_rol] as [ps].[id_rol]), WHERE:([open10].[dbo].[roles].[nombre] as [r].[nombre]='Dependiente') ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,004706
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1027],[open10].[dbo].[tickets].[id_persona] as [t].[id_persona],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
|--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	943224	0	3,800889	204,9957
|--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1022]))	Bitmap	Bitmap Create	813053	0	0,515569	28,40355
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	813053	0	0,515569	28,40355
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_id_ingreso] AS [i]), WHERE:([open10].[dbo].[ingresos].[fecha] as [i].[fecha]>='2013-01-01 00:00:00.000' AND [open10].[dbo].[ingresos].[fecha] as [i].[fecha]<='2013-01-31 00:00:00.000'))	Clustered Index Scan	Clustered Index Scan	813053	22,93794	2,750039	25,68798
|--Hash Match(Inner Join, HASH:([l].[id_ticket])=([im].[id_ingreso]))	Hash Match	Inner Join	943224	0	16,436750	172,7912
|--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	813033	0	0,020326	117,9428
| |--Compute Scalar(DEFINE:([Expr1012]=CONVERT_IMPLICIT(numeric(10,0),[open10].[dbo].[lineasticket].[unidades] as [l].[unidades]*[open10].[dbo].[lineasticket].[precio] as [l].[precio],0)*(0.12)))	Compute Scalar	Compute Scalar	813033	0	0,020326	117,9428
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	813033	0	1,132650	117,9225
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1022],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1023],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([im].[id_ingreso]))	Parallelism	Repartition Streams	9986423	0	7,905915	38,41167
|--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [im]), WHERE:([open10].[dbo].[lineasimpuesto].[monto] as [im].[monto]<(500) AND PROBE([Bitmap1028],[open10].[dbo].[lineasimpuesto].[id_ingreso] as [im].[id_ingreso],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	9986423	26,55572	2,750039	29,30576

ANEXO 66

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 1 SELECT USANDO ÍNDICE + JOIN 10 MILLONES DE DATOS..

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	983360	0	3,745971	200,6282
|--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	983360	0	2,300617	196,8822
|--Bitmap(HASH:([p].[id_producto]), DEFINE:([Opt_Bitmap1026]))	Bitmap	Bitmap Create	78763	0	0,158562	0,4493123
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	739422	0	2,155125	194,1323
|--Hash Match(Inner Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open10].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]=[open10].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]))	Hash Match	Inner Join	739422	0	2,050102	191,9771
|--Bitmap(HASH:([a].[id_atributoconjuntoinstancia]), DEFINE:([Opt_Bitmap1024]))	Bitmap	Bitmap Create	1106	0	0,030326	0,03897994
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,030326	0,03897994
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,000343	0,008653585
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	739422	0	1,377068	189,888
|--Hash Match(Inner Join, HASH:([t].[id_ticket])=([l].[id_ticket]))	Hash Match	Inner Join	739422	0	9,554862	188,511
|--Bitmap(HASH:([t].[id_ticket]), DEFINE:([Opt_Bitmap1023]))	Bitmap	Bitmap Create	3333334	0	2,025375	11,5685
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	3333334	0	2,025375	11,5685
| |--Nested Loops(Inner Join, OUTER REFERENCES:([ps].[id_persona]) OPTIMIZED)	Nested Loops	Inner Join	3333334	0	3,483334	9,543129
| |--Nested Loops(Inner Join, OUTER REFERENCES:([ps].[id_rol]))	Nested Loops	Inner Join	3	0	0,000010	0,03654058
| | |--Parallelism(Distribute Streams, RoundRobin Partitioning)	Parallelism	Distribute Streams	10	0	0,028526	0,03181932
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[persona].[persona_id_persona] AS [ps]))	Clustered Index Scan	Clustered Index Scan	10	0,003125	0,000168	0,003293
| | |--Clustered Index Seek(OBJECT:([open10].[dbo].[roles].[roles_id_rol] AS [r]), SEEK:([r].[id_rol]=[open10].[dbo].[persona].[id_rol] as [ps].[id_rol]), WHERE:([open10].[dbo].[roles].[nombre] as [r].[nombre]='Dependiente') ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,004706
| |--Index Seek(OBJECT:([open10].[dbo].[tickets].[Ix_persona_tickets>] AS [t]), SEEK:([t].[id_persona]=[open10].[dbo].[persona].[id_persona] as [ps].[id_persona]) ORDERED FORWARD)	Index Seek	Index Seek	1000000	1,009792	1,100157	6,023255
|--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	739422	0	3,496011	167,3876
|--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1022]))	Bitmap	Bitmap Create	847651	0	0,536296	1,937009
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	847651	0	0,536296	1,937009
| |--Index Seek(OBJECT:([open10].[dbo].[ingresos].[Ix_ingreso_fecha] AS [i]), SEEK:([i].[fecha] >= '2013-01-01 00:00:00.000' AND [i].[fecha] <= '2013-01-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	847651	1,167569	0,233143	1,400713
|--Hash Match(Inner Join, HASH:([l].[id_ticket])=([im].[id_ingreso]))	Hash Match	Inner Join	739422	0	15,361760	161,9546
|--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1032]))	Bitmap	Bitmap Create	637355	0	0,015934	122,1997
| |--Compute Scalar(DEFINE:([Expr1016]=CONVERT_IMPLICIT(numeric(10,0),[open10].[dbo].[lineasticket].[unidades] as [l].[unidades]*[open10].[dbo].[lineasticket].[precio] as [l].[precio],0)*(0.12)))	Compute Scalar	Compute Scalar	637355	0	0,015934	122,1997
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	637355	0	0,894068	122,1838
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1022],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1023],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1024],[open10].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia],N'[IN ROW]') AND PROBE([Opt_Bitmap1026],[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([im].[id_ingreso]))	Parallelism	Repartition Streams	9990574	0	7,909190	24,3931
|--Index Seek(OBJECT:([open10].[dbo].[lineasimpuesto].[limpuesto_monto] AS [im]), SEEK:([im].[monto] < (500)), WHERE:(PROBE([Bitmap1032],[open10].[dbo].[lineasimpuesto].[id_ingreso] as [im].[id_ingreso],N'[IN ROW]')) ORDERED FORWARD)	Index Seek	Index Seek	9990574	13,73646	2,747447	16,48391

ANEXO 67

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 1 SELECT CON ÍNDICE + SUBQUERY 10 MILLONES DE DATOS

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1031]=[open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]))	Compute Scalar	Compute Scalar	2540941	0	0,254094	300,561
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	2540941	0	5,838989	300,3069
|--Hash Match(Right Outer Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open10].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]=[open10].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Right Outer Join	2540941	0	7,112688	294,4679
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,030326	0,03897994
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,000343	0,008653585
|--Compute Scalar(DEFINE:([Expr1027]=[open10].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	2540941	0	0,063524	287,3163
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	2540941	0	6,854260	287,2527
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	2540941	0	5,543538	280,3985
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	2540941	0	4,180555	274,4056
|--Hash Match(Right Semi Join, HASH:([open10].[dbo].[tickets].[id_ticket])=([l].[id_ticket]))	Hash Match	Right Semi Join	2540941	0	11,634230	270,2251
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open10].[dbo].[tickets].[id_ticket]))	Parallelism	Repartition Streams	2774941	0	1,690863	29,51042
| |--Nested Loops(Inner Join, OUTER REFERENCES:([open10].[dbo].[persona].[id_persona]))	Nested Loops	Inner Join	2774941	0	2,899813	27,81956
| |--Sort(DISTINCT ORDER BY:([open10].[dbo].[persona].[id_persona] ASC))	Sort	Distinct Sort	3	0,002815315	0,000108	19,3212
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open10].[dbo].[persona].[id_persona]))	Parallelism	Repartition Streams	40	0	0,028510	19,31828
| | |--Hash Match(Partial Aggregate, HASH:([open10].[dbo].[persona].[id_persona]))	Hash Match	Partial Aggregate	40	0	0,341007	19,28977
| | |--Hash Match(Right Semi Join, HASH:([open10].[dbo].[ingresos].[id_ingreso])=([open10].[dbo].[tickets].[id_ticket]))	Hash Match	Right Semi Join	282550	0	6,112833	18,94876
| | |--Bitmap(HASH:([open10].[dbo].[ingresos].[id_ingreso]), DEFINE:([Bitmap1032]))	Bitmap	Bitmap Create	847651	0	0,536296	1,937009
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open10].[dbo].[ingresos].[id_ingreso]))	Parallelism	Repartition Streams	847651	0	0,536296	1,937009
| | | |--Index Seek(OBJECT:([open10].[dbo].[ingresos].[Ix_ingreso_fecha]), SEEK:([open10].[dbo].[ingresos].[fecha] >= '2013-01-01 00:00:00.000' AND [open10].[dbo].[ingresos].[fecha] <= '2013-01-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	847651	1,167569	0,233143	1,400713
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open10].[dbo].[tickets].[id_ticket]), WHERE:(PROBE([Bitmap1032],[open10].[dbo].[tickets].[id_ticket])))	Parallelism	Repartition Streams	3333334	0	2,657875	10,89892
| | |--Nested Loops(Inner Join, OUTER REFERENCES:([open10].[dbo].[persona].[id_persona]) OPTIMIZED)	Nested Loops	Inner Join	3333334	0	2,786535	8,241043
| | |--Nested Loops(Inner Join, OUTER REFERENCES:([open10].[dbo].[persona].[id_rol]))	Nested Loops	Inner Join	3	0	0,000010	0,03654058
| | | |--Parallelism(Distribute Streams, RoundRobin Partitioning)	Parallelism	Distribute Streams	10	0	0,028526	0,03181932
| | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[persona].[persona_id_persona]))	Clustered Index Scan	Clustered Index Scan	10	0,003125	0,000168	0,003293
| | | |--Clustered Index Seek(OBJECT:([open10].[dbo].[roles].[roles_id_rol]), SEEK:([open10].[dbo].[roles].[id_rol]=[open10].[dbo].[persona].[id_rol]), WHERE:([open10].[dbo].[roles].[nombre]='Dependiente') ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,004706
| | |--Index Seek(OBJECT:([open10].[dbo].[tickets].[Ix_persona_tickets>]), SEEK:([open10].[dbo].[tickets].[id_persona]=[open10].[dbo].[persona].[id_persona]) ORDERED FORWARD)	Index Seek	Index Seek	799962	0,8083102	0,880115	5,417968
| |--Index Seek(OBJECT:([open10].[dbo].[tickets].[Ix_persona_tickets>]), SEEK:([open10].[dbo].[tickets].[id_persona]=[open10].[dbo].[persona].[id_persona]) ORDERED FORWARD)	Index Seek	Index Seek	1000000	1,009792	1,100157	5,598541
|--Hash Match(Right Semi Join, HASH:([open10].[dbo].[lineasimpuesto].[id_ingreso])=([l].[id_ticket]))	Hash Match	Right Semi Join	2540941	0	38,672780	229,0804
|--Bitmap(HASH:([open10].[dbo].[lineasimpuesto].[id_ingreso]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	983384	0	0,000000	38,69828
| |--Hash Match(Inner Join, HASH:([open10].[dbo].[ingresos].[id_ingreso])=([open10].[dbo].[lineasimpuesto].[id_ingreso]))	Hash Match	Inner Join	983384	0	14,263890	38,69828
| |--Bitmap(HASH:([open10].[dbo].[ingresos].[id_ingreso]), DEFINE:([Bitmap1033]))	Bitmap	Bitmap Create	847651	0	0,536296	1,937009
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open10].[dbo].[ingresos].[id_ingreso]))	Parallelism	Repartition Streams	847651	0	0,536296	1,937009
| | |--Index Seek(OBJECT:([open10].[dbo].[ingresos].[Ix_ingreso_fecha]), SEEK:([open10].[dbo].[ingresos].[fecha] >= '2013-01-01 00:00:00.000' AND [open10].[dbo].[ingresos].[fecha] <= '2013-01-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	847651	1,167569	0,233143	1,400713
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open10].[dbo].[lineasimpuesto].[id_ingreso]))	Parallelism	Repartition Streams	9990574	0	6,013478	22,49738
| |--Index Seek(OBJECT:([open10].[dbo].[lineasimpuesto].[limpuesto_monto]), SEEK:([open10].[dbo].[lineasimpuesto].[monto] < (500)), WHERE:(PROBE([Bitmap1033],[open10].[dbo].[lineasimpuesto].[id_ingreso],N'[IN ROW]')) ORDERED FORWARD)	Index Seek	Index Seek	9990574	13,73646	2,747447	16,48391
|--Compute Scalar(DEFINE:([Expr1023]=CONVERT_IMPLICIT(numeric(10,0),[open10].[dbo].[lineasticket].[unidades] as [l].[unidades]*[open10].[dbo].[lineasticket].[precio] as [l].[precio],0)*(0.12)))	Compute Scalar	Compute Scalar	29999270	0	0,749982	151,7094
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	29999270	0	40,769380	150,9594
|--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Bitmap1034],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19

ANEXO 68

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 1 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1020]=[open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]))	Compute Scalar	Compute Scalar	983360	0	0,098336	198,7423
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	983360	0	3,541650	198,6439
|--Hash Match(Right Outer Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open10].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]=[open10].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Right Outer Join	983360	0	2,716128	195,1023
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,030326	0,03897994
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,000343	0,008653585
|--Compute Scalar(DEFINE:([Expr1016]=[open10].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	983360	0	0,024584	192,3472
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	983360	0	2,856706	192,3226
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	983360	0	2,568949	189,4659
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	983360	0	1,821965	186,4476
|--Hash Match(Inner Join, HASH:([t].[id_ticket])=([l].[id_ticket]))	Hash Match	Inner Join	983360	0	10,020720	184,6256
|--Bitmap(HASH:([t].[id_ticket]), DEFINE:([Opt_Bitmap1023]))	Bitmap	Bitmap Create	3333334	0	2,025375	9,668131
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	3333334	0	2,025375	9,668131
| |--Nested Loops(Inner Join, OUTER REFERENCES:([ps].[id_persona]) OPTIMIZED)	Nested Loops	Inner Join	3333334	0	2,786535	7,642756
| |--Nested Loops(Left Semi Join, WHERE:([open10].[dbo].[persona].[id_rol] as [ps].[id_rol]=[open10].[dbo].[roles].[id_rol] as [r].[id_rol]))	Nested Loops	Left Semi Join	3	0	0,000010	0,03653878
| | |--Parallelism(Distribute Streams, RoundRobin Partitioning)	Parallelism	Distribute Streams	10	0	0,028526	0,03181932
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[persona].[persona_id_persona] AS [ps]))	Clustered Index Scan	Clustered Index Scan	10	0,003125	0,000168	0,003293
| | |--Index Seek(OBJECT:([open10].[dbo].[roles].[roles_nombre] AS [r]), SEEK:([r].[nombre]='Dependiente') ORDERED FORWARD)	Index Seek	Index Seek	1	0,003125	0,000158	0,004706
| |--Index Seek(OBJECT:([open10].[dbo].[tickets].[Ix_persona_tickets>] AS [t]), SEEK:([t].[id_persona]=[open10].[dbo].[persona].[id_persona] as [ps].[id_persona]) ORDERED FORWARD)	Index Seek	Index Seek	799962	0,8083102	0,880115	4,819683
|--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	983360	0	3,961872	164,9368
|--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1022]))	Bitmap	Bitmap Create	847651	0	0,536296	1,937009
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	847651	0	0,536296	1,937009
| |--Index Seek(OBJECT:([open10].[dbo].[ingresos].[Ix_ingreso_fecha] AS [i]), SEEK:([i].[fecha] >= '2013-01-01 00:00:00.000' AND [i].[fecha] <= '2013-01-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	847651	1,167569	0,233143	1,400713
|--Hash Match(Inner Join, HASH:([l].[id_ticket])=([im].[id_ingreso]))	Hash Match	Inner Join	983360	0	16,654140	159,0379
|--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1027]))	Bitmap	Bitmap Create	847630	0	0,021191	117,9907
| |--Compute Scalar(DEFINE:([Expr1012]=CONVERT_IMPLICIT(numeric(10,0),[open10].[dbo].[lineasticket].[unidades] as [l].[unidades]*[open10].[dbo].[lineasticket].[precio] as [l].[precio],0)*(0.12)))	Compute Scalar	Compute Scalar	847630	0	0,021191	117,9907
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	847630	0	1,179635	117,9695
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1022],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1023],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([im].[id_ingreso]))	Parallelism	Repartition Streams	9990574	0	7,909190	24,3931
|--Index Seek(OBJECT:([open10].[dbo].[lineasimpuesto].[limpuesto_monto] AS [im]), SEEK:([im].[monto] < (500)), WHERE:(PROBE([Bitmap1027],[open10].[dbo].[lineasimpuesto].[id_ingreso] as [im].[id_ingreso],N'[IN ROW]')) ORDERED FORWARD)	Index Seek	Index Seek	9990574	13,73646	2,747447	16,48391

ANEXO 69

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 2 SELECT SIMPLE Y SELECT USANDO JOIN 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	67810	0	0,4649863	206,5971
|--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	67810	0	0,5579013	206,1321
|--Bitmap(HASH:([p].[id_producto]), DEFINE:([Opt_Bitmap1016]))	Bitmap	Bitmap Create	39382	0	0,1213837	0,8698564
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	39382	0	0,1213837	0,8698564
| |--Nested Loops(Inner Join, WHERE:([open10].[dbo].[categorias].[id_categoria] as [c].[id_categoria]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Nested Loops	Inner Join	39382	0	0,08230734	0,7484726
| |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	1	0	0,02850332	0,03178642
| | |--Index Seek(OBJECT:([open10].[dbo].[categorias].[categorias_nombre] AS [c]), SEEK:([c].[nombre]='FARMACEUTICOS') ORDERED FORWARD)	Index Seek	Index Seek	1	0,003125	0,0001581	0,0032831
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,02169907	0,61075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	67810	0	0,3557058	204,7043
|--Hash Match(Inner Join, HASH:([aci].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open10].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [aci].[id_atributoconjuntoinstancia]=[open10].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Inner Join	67810	0	0,2029537	204,3486
|--Bitmap(HASH:([aci].[id_atributoconjuntoinstancia]), DEFINE:([Opt_Bitmap1015]))	Bitmap	Bitmap Create	5	0	0,0285242	0,04017667
| |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	5	0	0,0285242	0,04017667
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [aci]), WHERE:([open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO NATURPHARMA' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO VITAFARMA'))	Clustered Index Scan	Clustered Index Scan	5	0,008310185	0,0013736	0,009683785
|--Hash Match(Inner Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]=[open10].[dbo].[tickets].[id_cliente] as [t].[id_cliente]))	Hash Match	Inner Join	67810	0	4,171535	204,1055
|--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1019]))	Bitmap	Bitmap Create	67810	0	0,1205906	162,3916
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	67810	0	0,1205906	162,3916
| |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	67810	0	11,76809	162,271
| |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1018]))	Bitmap	Bitmap Create	67810	0	0,1077236	116,8976
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	67810	0	0,1077236	116,8976
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1015],[open10].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia],N'[IN ROW]') AND PROBE([Opt_Bitmap1016],[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	7,916625	33,60535
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1018],[open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	7,555548	37,54234
|--Clustered Index Scan(OBJECT:([open10].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1019],[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,37053	0,6162642	29,9868

ANEXO 70

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 2 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	67810	0	0,4649863	92,59272
|--Hash Match(Inner Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]=[open10].[dbo].[tickets].[id_cliente] as [t].[id_cliente]))	Hash Match	Inner Join	67810	0	5,537427	92,12773
|--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1019]))	Bitmap	Bitmap Create	67810	0	0,2890314	49,04796
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	67810	0	0,2890314	49,04796
| |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	67810	0	12,72587	48,75893
| |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1018]))	Bitmap	Bitmap Create	67903	0	0,2765013	2,427705
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	67903	0	0,2765013	2,427705
| | |--Hash Match(Inner Join, HASH:([p].[id_producto])=([ll].[id_producto]))	Hash Match	Inner Join	67903	0	0,5579759	2,151204
| | |--Bitmap(HASH:([p].[id_producto]), DEFINE:([Opt_Bitmap1014]))	Bitmap	Bitmap Create	39382	0	0,1213837	0,8698564
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	39382	0	0,1213837	0,8698564
| | | |--Nested Loops(Inner Join, WHERE:([open10].[dbo].[categorias].[id_categoria] as [c].[id_categoria]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Nested Loops	Inner Join	39382	0	0,08230734	0,7484726
| | | | |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	1	0	0,02850332	0,03178642
| | | | | |--Index Seek(OBJECT:([open10].[dbo].[categorias].[categorias_nombre] AS [c]), SEEK:([c].[nombre]='FARMACEUTICOS') ORDERED FORWARD)	Index Seek	Index Seek	1	0,003125	0,0001581	0,0032831
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,02169907	0,61075
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	67810	0	0,1580219	0,7233685
| | |--Nested Loops(Inner Join, OUTER REFERENCES:([aci].[id_atributoconjuntoinstancia]))	Nested Loops	Inner Join	67810	0	0,07086175	0,5653466
| | |--Parallelism(Repartition Streams, RoundRobin Partitioning)	Parallelism	Repartition Streams	5	0	0,02850605	0,03765181
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [aci]), WHERE:([open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO NATURPHARMA' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO VITAFARMA'))	Clustered Index Scan	Clustered Index Scan	5	0,008310185	0,0003434	0,008653585
| | | |--Index Seek(OBJECT:([open10].[dbo].[lineasticket].[atribconjinst_descrip] AS [l]), SEEK:([l].[id_atributoconjuntoinstancia]=[open10].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [aci].[id_atributoconjuntoinstancia]), WHERE:(PROBE([Opt_Bitmap1014],[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto],N'[IN ROW]')) ORDERED FORWARD)	Index Seek	Index Seek	27124	0,06756944	0,02999352	0,4202412
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	7,916625	33,60535
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1018],[open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	7,555548	37,54234
|--Clustered Index Scan(OBJECT:([open10].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1019],[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,37053	0,6162642	29,9868

ANEXO 71

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 2 SELECT USANDO SUBQUERY 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1017]=[Expr1031], [Expr1029]=[open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]))	Compute Scalar	Compute Scalar	67810	0	0,006781	299,9347
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	67810	0	0,440821	299,9279
|--Hash Match(Right Outer Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]))	Hash Match	Right Outer Join	67810	0	0,156803	299,4871
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,029983	0,03863652
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,000343	0,008653585
|--Compute Scalar(DEFINE:([Expr1025]=[open10].[dbo].[productos].[precioventa] as [p].[precioventa]))	Compute Scalar	Compute Scalar	67810	0	0,001695	299,2917
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	67810	0	0,422380	299,29
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	67810	0	0,376254	298,8676
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,090629	0,7013792
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
|--Compute Scalar(DEFINE:([Expr1021]=[open10].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	67810	0	0,001695	297,79
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	67810	0	0,820477	297,7883
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Filter(WHERE:([Expr1031]='FARMACEUTICOS'))	Filter	Filter	67810	0	0,016274	296,5185
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	135621	0	0,652499	296,5022
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	135621	0	0,141724	295,8497
|--Compute Scalar(DEFINE:([Expr1011]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	135621	0	0,003391	249,5655
| |--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	135621	0	0,505755	249,5621
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,090629	0,7013792
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	135621	0	0,586994	248,3549
| |--Hash Match(Inner Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open10].[dbo].[tickets].[id_cliente] as [t].[id_cliente]=[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]))	Hash Match	Inner Join	135621	0	4,663982	247,7679
| |--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1036]))	Bitmap	Bitmap Create	135621	0	0,000000	205,5616
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	135621	0	0,212681	205,5616
| | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	135621	0	12,080940	205,3489
| | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1035]))	Bitmap	Bitmap Create	135621	0	0,186947	159,6626
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	135621	0	0,186947	159,6626
| | | |--Hash Match(Inner Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open10].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]=[open10].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]))	Hash Match	Inner Join	135621	0	49,245530	159,4757
| | | |--Bitmap(HASH:([a].[id_atributoconjuntoinstancia]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	5	0	0,028509	0,04016183
| | | | |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	5	0	0,028509	0,04016183
| | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]), WHERE:([open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO NATURPHARMA' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO VITAFARMA'))	Clustered Index Scan	Clustered Index Scan	5	0,008310185	0,001374	0,009683785
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Bitmap1034],[open10].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	7,916625	33,60535
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1035],[open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	7,555548	37,54234
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1036],[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,37053	0,616264	29,9868
|--Assert(WHERE:(CASE WHEN [Expr1030]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	46,14252
|--Stream Aggregate(DEFINE:([Expr1030]=Count(*), [Expr1031]=ANY([open10].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	0,000001	46,07742
|--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,000004	45,92824
|--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	23,91661
|--Clustered Index Seek(OBJECT:([open10].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	21,44474

ANEXO 72

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 2 SELECT USANDO SUBQUERY + JOIN 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1017]=[Expr1027], [Expr1025]=[open10].[dbo].[productos].[precioventa] as [p].[precioventa]))	Compute Scalar	Compute Scalar	67810	0	0,006781	254,2702
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	67810	0	0,461477	254,2634
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	67810	0	0,376254	253,8019
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,090629	0,7013792
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
|--Compute Scalar(DEFINE:([Expr1021]=[open10].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	67810	0	0,001695	252,7243
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	67810	0	0,820477	252,7226
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Filter(WHERE:([Expr1027]='FARMACEUTICOS'))	Filter	Filter	67810	0	0,016274	251,4528
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	135621	0	0,748416	251,4365
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	135621	0	0,141724	250,6881
|--Compute Scalar(DEFINE:([Expr1011]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	135621	0	0,003391	204,4039
| |--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	135621	0	0,505755	204,4005
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,090629	0,7013792
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	135621	0	0,682912	203,1933
| |--Hash Match(Inner Join, HASH:([aci].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open10].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [aci].[id_atributoconjuntoinstancia]=[open10].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Inner Join	135621	0	0,388096	202,5104
| |--Bitmap(HASH:([aci].[id_atributoconjuntoinstancia]), DEFINE:([Opt_Bitmap1028]))	Bitmap	Bitmap Create	5	0	0,028524	0,04017667
| | |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	5	0	0,028524	0,04017667
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [aci]), WHERE:([open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO NATURPHARMA' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO VITAFARMA'))	Clustered Index Scan	Clustered Index Scan	5	0,008310185	0,001374	0,009683785
| |--Hash Match(Inner Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]=[open10].[dbo].[tickets].[id_cliente] as [t].[id_cliente]))	Hash Match	Inner Join	135621	0	4,663982	202,0822
| |--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	135621	0	0,212681	159,8758
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	135621	0	0,212681	159,8758
| | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	135621	0	12,080940	159,6631
| | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1033]))	Bitmap	Bitmap Create	135621	0	0,186947	113,9769
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	135621	0	0,186947	113,9769
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1028],[open10].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	7,916625	33,60535
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1033],[open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	7,555548	37,54234
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1034],[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,37053	0,616264	29,9868
|--Assert(WHERE:(CASE WHEN [Expr1026]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	46,14252
|--Stream Aggregate(DEFINE:([Expr1026]=Count(*), [Expr1027]=ANY([open10].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	0,000001	46,07742
|--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,000004	45,92824
|--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	23,91661
|--Clustered Index Seek(OBJECT:([open10].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	21,44474

ANEXO 73

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 2 SELECT CON ÍNDICE + SUBQUERY 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1017]=[Expr1031], [Expr1029]=[open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]))	Compute Scalar	Compute Scalar	67810	0	0,006781	141,0312
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	67810	0	0,441952	141,0244
|--Hash Match(Right Outer Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]))	Hash Match	Right Outer Join	67810	0	0,156722	140,5825
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,029983	0,03863652
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,000343	0,008653585
|--Compute Scalar(DEFINE:([Expr1025]=[open10].[dbo].[productos].[precioventa] as [p].[precioventa]))	Compute Scalar	Compute Scalar	67810	0	0,001695	140,3871
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	67810	0	0,422380	140,3854
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	67810	0	0,376254	139,963
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,090629	0,7013792
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
|--Compute Scalar(DEFINE:([Expr1021]=[open10].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	67810	0	0,001695	138,8854
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	67810	0	0,820477	138,8837
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Filter(WHERE:([Expr1031]='FARMACEUTICOS'))	Filter	Filter	67810	0	0,016274	137,6139
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	135621	0	0,652499	137,5976
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	135621	0	0,141724	136,9451
|--Compute Scalar(DEFINE:([Expr1011]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	135621	0	0,003391	90,66087
| |--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	135621	0	0,505755	90,65749
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,090629	0,7013792
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	135621	0	0,586994	89,45035
| |--Hash Match(Inner Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open10].[dbo].[tickets].[id_cliente] as [t].[id_cliente]=[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]))	Hash Match	Inner Join	135621	0	4,663982	88,86335
| |--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1035]))	Bitmap	Bitmap Create	135621	0	0,000000	46,65702
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	135621	0	0,212681	46,65702
| | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	135621	0	12,080940	46,44434
| | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	135621	0	0,186947	0,7580577
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	135621	0	0,186947	0,7580577
| | | |--Nested Loops(Inner Join, OUTER REFERENCES:([a].[id_atributoconjuntoinstancia]))	Nested Loops	Inner Join	135621	0	0,141724	0,5711104
| | | |--Parallelism(Repartition Streams, RoundRobin Partitioning)	Parallelism	Repartition Streams	5	0	0,028502	0,0376481
| | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]), WHERE:([open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO NATURPHARMA' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO VITAFARMA'))	Clustered Index Scan	Clustered Index Scan	5	0,008310185	0,000343	0,008653585
| | | |--Index Seek(OBJECT:([open10].[dbo].[lineasticket].[atribconjinst_descrip] AS [l]), SEEK:([l].[id_atributoconjuntoinstancia]=[open10].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]) ORDERED FORWARD)	Index Seek	Index Seek	27124	0,06756944	0,029994	0,4202412
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	7,916625	33,60535
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1034],[open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	7,555548	37,54234
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1035],[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW')))	Clustered Index Scan	Clustered Index Scan	2240818	29,37053	0,616264	29,9868
|--Assert(WHERE:(CASE WHEN [Expr1030]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	46,14252
|--Stream Aggregate(DEFINE:([Expr1030]=Count(*), [Expr1031]=ANY([open10].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	0,000001	46,07742
|--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,000004	45,92824
|--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	23,91661
|--Clustered Index Seek(OBJECT:([open10].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	21,44474

ANEXO 74

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 2 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1017]=[Expr1027], [Expr1025]=[open10].[dbo].[productos].[precioventa] as [p].[precioventa]))	Compute Scalar	Compute Scalar	67810	0	0,006781	142,8367
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	67810	0	0,461477	142,8299
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	67810	0	0,376254	142,3685
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,090629	0,7013792
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
|--Compute Scalar(DEFINE:([Expr1021]=[open10].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	67810	0	0,001695	141,2908
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	67810	0	0,820477	141,2891
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Filter(WHERE:([Expr1027]='FARMACEUTICOS'))	Filter	Filter	67810	0	0,016274	140,0193
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	135621	0	0,748416	140,0031
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	135621	0	0,141724	139,2546
|--Compute Scalar(DEFINE:([Expr1011]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	135621	0	0,003391	92,9704
| |--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	135621	0	0,505755	92,96701
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,090629	0,7013792
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	135621	0	0,682912	91,75987
| |--Hash Match(Inner Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]=[open10].[dbo].[tickets].[id_cliente] as [t].[id_cliente]))	Hash Match	Inner Join	135621	0	5,847756	91,07696
| |--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	135621	0	0,313278	47,68686
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	135621	0	0,313278	47,68686
| | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	135621	0	12,909580	47,37358
| | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1033]))	Bitmap	Bitmap Create	135621	0	0,287544	0,8586542
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	135621	0	0,287544	0,8586542
| | | |--Nested Loops(Inner Join, OUTER REFERENCES:([aci].[id_atributoconjuntoinstancia]))	Nested Loops	Inner Join	135621	0	0,141724	0,5711104
| | | |--Parallelism(Repartition Streams, RoundRobin Partitioning)	Parallelism	Repartition Streams	5	0	0,028506	0,03765181
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [aci]), WHERE:([open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO NATURPHARMA' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR [open10].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO VITAFARMA'))	Clustered Index Scan	Clustered Index Scan	5	0,008310185	0,000343	0,008653585
| | | |--Index Seek(OBJECT:([open10].[dbo].[lineasticket].[atribconjinst_descrip] AS [l]), SEEK:([l].[id_atributoconjuntoinstancia]=[open10].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [aci].[id_atributoconjuntoinstancia]) ORDERED FORWARD)	Index Seek	Index Seek	27124	0,06756944	0,029994	0,4202412
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	7,916625	33,60535
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1033],[open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	7,555548	37,54234
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1034],[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW'])))	Clustered Index Scan	Clustered Index Scan	2240818	29,37053	0,616264	29,9868
|--Assert(WHERE:(CASE WHEN [Expr1026]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	46,14252
|--Stream Aggregate(DEFINE:([Expr1026]=Count(*), [Expr1027]=ANY([open10].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	0,000001	46,07742
|--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,000004	45,92824
|--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	23,91661
|--Clustered Index Seek(OBJECT:([open10].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	21,44474

ANEXO 75

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 3 SELECT SIMPLE Y SELECT USANDO JOIN 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,028510	771,3419
|--Filter(WHERE:([Expr1010]>=(600) AND [Expr1010]<=(1000)))	Filter	Filter	1	0	2,068256	771,3134
|--Stream Aggregate(GROUP BY:([l].[id_producto], [l].[precio], [c].[nombre]) DEFINE:([Expr1010]=SUM([open10].[dbo].[lineasticket].[unidades] as [l].[unidades]), [p].[nombre]=ANY([open10].[dbo].[productos].[nombre] as [p].[nombre]), [c].[id_categoria]=ANY([open10].[dbo].[categorias].[id_categoria] as [c].[id_categoria]))	Stream Aggregate	Aggregate	9401165	0	2,672672	769,2451
|--Sort(ORDER BY:([l].[id_producto] ASC, [l].[precio] ASC, [c].[nombre] ASC))	Sort	Sort	9983506	401,5259	121,125600	766,5724
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto], [l].[precio], [c].[nombre]))	Parallelism	Repartition Streams	9983506	0	27,399530	243,921
|--Hash Match(Inner Join, HASH:([p].[id_producto])=([ll].[id_producto]))	Hash Match	Inner Join	9983506	0	19,868030	216,5214
|--Bitmap(HASH:([p].[id_producto]), DEFINE:([Opt_Bitmap1012]))	Bitmap	Bitmap Create	78763	0	0,370175	1,267548
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,370175	1,267548
| |--Hash Match(Inner Join, HASH:([c].[id_categoria])=([p].[id_categoria]))	Hash Match	Inner Join	78763	0	0,254829	0,897373
| |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	2	0	0,028507	0,03179085
| | |--Index Scan(OBJECT:([open10].[dbo].[categorias].[categorias_nombre] AS [c]))	Index Scan	Index Scan	2	0,003125	0,000159	0,0032842
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	9983506	0	9,797985	195,3858
|--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	9983506	0	27,195550	185,5879
|--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1011]))	Bitmap	Bitmap Create	3327917	0	2,022130	29,91011
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	3327917	0	2,022130	29,91011
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_id_ingreso] AS [i]), WHERE:([open10].[dbo].[ingresos].[fecha] as [i].[fecha]>='2013-05-01 00:00:00.000' AND [open10].[dbo].[ingresos].[fecha] as [i].[fecha]<='2013-08-31 00:00:00.000'))	Clustered Index Scan	Clustered Index Scan	3327917	22,93794	2,750039	25,68798
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	9983506	0	11,692360	128,4822
|--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1011],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1012],[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto],N'[IN ROW]'))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19

ANEXO 76

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 3 SELECT ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,02850967	746,9838
|--Filter(WHERE:([Expr1010]>=(600) AND [Expr1010]<=(1000)))	Filter	Filter	1	0	2,06237	746,9553
|--Stream Aggregate(GROUP BY:([l].[id_producto], [l].[precio], [c].[nombre]) DEFINE:([Expr1010]=SUM([open10].[dbo].[lineasticket].[unidades] as [l].[unidades]), [p].[nombre]=ANY([open10].[dbo].[productos].[nombre] as [p].[nombre]), [c].[id_categoria]=ANY([open10].[dbo].[categorias].[id_categoria] as [c].[id_categoria]))	Stream Aggregate	Aggregate	9374408	0	2,664778	744,8929
|--Sort(ORDER BY:([l].[id_producto] ASC, [l].[precio] ASC, [c].[nombre] ASC))	Sort	Sort	9953178	400,3069	120,7075	742,2281
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto], [l].[precio], [c].[nombre]))	Parallelism	Repartition Streams	9953178	0	27,31638	221,2137
|--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	9953178	0	19,81011	193,8974
|--Bitmap(HASH:([p].[id_producto]), DEFINE:([Opt_Bitmap1012]))	Bitmap	Bitmap Create	78763	0	0,3701747	1,267548
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,3701747	1,267548
| |--Hash Match(Inner Join, HASH:([c].[id_categoria])=([p].[id_categoria]))	Hash Match	Inner Join	78763	0	0,2548291	0,897373
| |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	2	0	0,02850664	0,03179085
| | |--Index Scan(OBJECT:([open10].[dbo].[categorias].[categorias_nombre] AS [c]))	Index Scan	Index Scan	2	0,003125	0,0001592	0,0032842
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,02169907	0,61075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	9953178	0	9,768307	172,8197
|--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	9953178	0	27,11299	163,0514
|--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1011]))	Bitmap	Bitmap Create	3317807	0	2,016074	7,491635
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	3317807	0	2,016074	7,491635
| |--Index Seek(OBJECT:([open10].[dbo].[ingresos].[ingresos_fecha] AS [i]), SEEK:([i].[fecha] >= '2013-05-01 00:00:00.000' AND [i].[fecha] <= '2013-08-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	3317807	4,563125	0,9124362	5,475561
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	9953178	0	11,65692	128,4468
|--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1011],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1012],[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto],N'[IN ROW]'))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19

ANEXO 77

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 3 SELECT SUBQUERY 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1018]=[Expr1024], [Expr1022]=[open10].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	1	0	0,000000	368,7626
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,028509	368,7626
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,000001	368,7341
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,000001	368,7308
| |--Compute Scalar(DEFINE:([Expr1012]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	1	0	0,000000	368,7243
| | |--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,000001	368,7243
| | |--Filter(WHERE:([Expr1008]>=(600) AND [Expr1008]<=(1000)))	Filter	Filter	1	0	1,833332	368,721
| | | |--Hash Match(Aggregate, HASH:([l].[id_producto], [l].[precio]), RESIDUAL:([open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto] = [open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto] AND [open10].[dbo].[lineasticket].[precio] as [l].[precio] = [open10].[dbo].[lineasticket].[precio] as [l].[precio]) DEFINE:([Expr1008]=SUM([open10].[dbo].[lineasticket].[unidades] as [l].[unidades])))	Hash Match	Aggregate	8333327	16,51575	73,730770	366,8877
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto], [l].[precio]))	Parallelism	Repartition Streams	9983506	0	10,423200	276,6411
| | | |--Hash Match(Inner Join, HASH:([open10].[dbo].[ingresos].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	9983506	0	50,088580	266,2179
| | | |--Bitmap(HASH:([open10].[dbo].[ingresos].[id_ingreso]), DEFINE:([Bitmap1029]))	Bitmap	Bitmap Create	3327917	0	22,116230	70,86234
| | | | |--Hash Match(Inner Join, HASH:([open10].[dbo].[ingresos].[id_ingreso])=([open10].[dbo].[tickets].[id_ticket]))	Hash Match	Inner Join	3327917	0	22,116230	70,86234
| | | | |--Bitmap(HASH:([open10].[dbo].[ingresos].[id_ingreso]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	3327917	0	2,022130	29,91011
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open10].[dbo].[ingresos].[id_ingreso]))	Parallelism	Repartition Streams	3327917	0	2,022130	29,91011
| | | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_id_ingreso]), WHERE:([open10].[dbo].[ingresos].[fecha]>='2013-05-01 00:00:00.000' AND [open10].[dbo].[ingresos].[fecha]<='2013-08-31 00:00:00.000'))	Clustered Index Scan	Clustered Index Scan	3327917	22,93794	2,750039	25,68798
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open10].[dbo].[tickets].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	6,019125	18,83599
| | | | |--Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_ticketipo]), WHERE:(PROBE([Bitmap1028],[open10].[dbo].[tickets].[id_ticket],N'[IN ROW]')))	Index Scan	Index Scan	10000000	10,06683	2,750039	12,81687
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	29999270	0	35,077020	145,267
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Bitmap1029],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
| | |--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
| |--Assert(WHERE:(CASE WHEN [Expr1023]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	0,00657196
| |--Stream Aggregate(DEFINE:([Expr1023]=Count(*), [Expr1024]=ANY([open10].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	0,000001	0,00657148
| |--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,000004	0,00657038
| |--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
| |--Clustered Index Seek(OBJECT:([open10].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
|--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831

ANEXO 78

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 3 SELECT USANDO SUBQUERY + JOIN 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1018]=[Expr1020]))	Compute Scalar	Compute Scalar	1	0	0,000000	532,6511
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,028508	532,6511
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,000001	532,6226
|--Compute Scalar(DEFINE:([Expr1012]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	1	0	0,000000	532,616
| |--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,000001	532,616
| |--Filter(WHERE:([Expr1008]>=(600) AND [Expr1008]<=(1000)))	Filter	Filter	1	0	1,833332	532,6127
| | |--Hash Match(Aggregate, HASH:([p].[nombre], [l].[precio]), RESIDUAL:([open10].[dbo].[productos].[nombre] as [p].[nombre] = [open10].[dbo].[productos].[nombre] as [p].[nombre] AND [open10].[dbo].[lineasticket].[precio] as [l].[precio] = [open10].[dbo].[lineasticket].[precio] as [l].[precio]) DEFINE:([Expr1008]=SUM([open10].[dbo].[lineasticket].[unidades] as [l].[unidades]), [l].[id_producto]=ANY([open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]))	Hash Match	Aggregate	8333327	146,7198	148,948200	530,7794
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[nombre], [l].[precio]))	Parallelism	Repartition Streams	9983506	0	22,553160	235,1114
| | |--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	9983506	0	19,756980	212,5582
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,124596	0,4153458
| | | |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	9983506	0	9,797985	192,3859
| | |--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	9983506	0	27,195550	182,5879
| | |--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1021]))	Bitmap	Bitmap Create	3327917	0	2,022130	29,91011
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	3327917	0	2,022130	29,91011
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_id_ingreso] AS [i]), WHERE:([open10].[dbo].[ingresos].[fecha] as [i].[fecha]>='2013-05-01 00:00:00.000' AND [open10].[dbo].[ingresos].[fecha] as [i].[fecha]<='2013-08-31 00:00:00.000'))	Clustered Index Scan	Clustered Index Scan	3327917	22,93794	2,750039	25,68798
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	9983506	0	11,692360	125,4823
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1021],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
| |--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
|--Assert(WHERE:(CASE WHEN [Expr1019]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	0,00657196
|--Stream Aggregate(DEFINE:([Expr1019]=Count(*), [Expr1020]=ANY([open10].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	0,000001	0,00657148
|--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,000004	0,00657038
|--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
|--Clustered Index Seek(OBJECT:([open10].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831

. ANEXO 79

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 3 SELECT CON ÍNDICE + SUBQUERY 10 MILLONES DE DATOS

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1022]=[Expr1024]))	Compute Scalar	Compute Scalar	1	0	0,000000	345,829
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,028509	345,829
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,000001	345,8005
|--Compute Scalar(DEFINE:([Expr1016]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	1	0	0,000000	345,794
| |--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,000001	345,794
| |--Compute Scalar(DEFINE:([Expr1012]=[open10].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	1	0	0,000000	345,7907
| | |--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,000001	345,7907
| | |--Filter(WHERE:([Expr1008]>=(600) AND [Expr1008]<=(1000)))	Filter	Filter	1	0	1,828742	345,7874
| | | |--Hash Match(Aggregate, HASH:([l].[id_producto], [l].[precio]), RESIDUAL:([open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto] = [open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto] AND [open10].[dbo].[lineasticket].[precio] as [l].[precio] = [open10].[dbo].[lineasticket].[precio] as [l].[precio]) DEFINE:([Expr1008]=SUM([open10].[dbo].[lineasticket].[unidades] as [l].[unidades]))	Hash Match	Aggregate	8312462	16,31827	73,529550	343,9586
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto], [l].[precio]))	Parallelism	Repartition Streams	9953178	0	10,391620	254,1108
| | | |--Hash Match(Inner Join, HASH:([open10].[dbo].[ingresos].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	9953178	0	50,040700	243,7192
| | | |--Bitmap(HASH:([open10].[dbo].[ingresos].[id_ingreso]), DEFINE:([Bitmap1029]))	Bitmap	Bitmap Create	3317807	0	22,083850	48,41148
| | | | |--Hash Match(Inner Join, HASH:([open10].[dbo].[ingresos].[id_ingreso])=([open10].[dbo].[tickets].[id_ticket]))	Hash Match	Inner Join	3317807	0	22,083850	48,41148
| | | | |--Bitmap(HASH:([open10].[dbo].[ingresos].[id_ingreso]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	3317807	0	2,016074	7,491635
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open10].[dbo].[ingresos].[id_ingreso]))	Parallelism	Repartition Streams	3317807	0	2,016074	7,491635
| | | | | |--Index Seek(OBJECT:([open10].[dbo].[ingresos].[ingresos_fecha]), SEEK:([open10].[dbo].[ingresos].[fecha] >= '2013-05-01 00:00:00.000' AND [open10].[dbo].[ingresos].[fecha] <= '2013-08-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	3317807	4,563125	0,912436	5,475561
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open10].[dbo].[tickets].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	6,019125	18,83599
| | | | |--Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_ticketipo]), WHERE:(PROBE([Bitmap1028],[open10],[dbo],[tickets],[id_ticket],N'[IN ROW]')))	Index Scan	Index Scan	10000000	10,06683	2,750039	12,81687
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	29999270	0	35,077020	145,267
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Bitmap1029],[open10],[dbo],[lineasticket],[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
| | |--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
| |--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
|--Assert(WHERE:(CASE WHEN [Expr1023]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	0,00657196
|--Stream Aggregate(DEFINE:([Expr1023]=Count(*), [Expr1024]=ANY([open10].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	0,000001	0,00657148
|--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,000004	0,00657038
|--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
|--Clustered Index Seek(OBJECT:([open10].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831

ANEXO 80

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 3 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1018]=[Expr1020]))	Compute Scalar	Compute Scalar	1	0	0,000000	508,9639
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,028508	508,9639
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,000001	508,9354
|--Compute Scalar(DEFINE:([Expr1012]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	1	0	0,000000	508,9289
| |--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,000001	508,9289
| |--Filter(WHERE:([Expr1008]>=(600) AND [Expr1008]<=(1000)))	Filter	Filter	1	0	1,828742	508,9256
| | |--Hash Match(Aggregate, HASH:([p].[nombre], [l].[precio]), RESIDUAL:([open10].[dbo].[productos].[nombre] as [p].[nombre] = [open10].[dbo].[productos].[nombre] as [p].[nombre] AND [open10].[dbo].[lineasticket].[precio] as [l].[precio] = [open10].[dbo].[lineasticket].[precio] as [l].[precio]) DEFINE:([Expr1008]=SUM([open10].[dbo].[lineasticket].[unidades] as [l].[unidades]), [l].[id_producto]=ANY([open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto])))	Hash Match	Aggregate	8312462	146,1267	148,551200	507,0968
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[nombre], [l].[precio]))	Parallelism	Repartition Streams	9953178	0	22,484740	212,4189
| | |--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	9953178	0	19,699060	189,9342
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,124596	0,4153458
| | | |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	9953178	0	9,768307	169,8198
| | |--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	9953178	0	27,112990	160,0515
| | |--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1021]))	Bitmap	Bitmap Create	3317807	0	2,016074	7,491635
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	3317807	0	2,016074	7,491635
| | | |--Index Seek(OBJECT:([open10].[dbo].[ingresos].[ingresos_fecha] AS [i]), SEEK:([i].[fecha] >= '2013-05-01 00:00:00.000' AND [i].[fecha] <= '2013-08-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	3317807	4,563125	0,912436	5,475561
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	9953178	0	11,656920	125,4468
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1021],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
| |--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
|--Assert(WHERE:(CASE WHEN [Expr1019]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	0,00657196
|--Stream Aggregate(DEFINE:([Expr1019]=Count(*), [Expr1020]=ANY([open10].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	0,000001	0,00657148
|--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,000004	0,00657038
|--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
|--Clustered Index Seek(OBJECT:([open10].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open10].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831

ANEXO 81

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 4 SELECT SIMPLE 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,028517	304,218
|--Nested Loops(Inner Join, OUTER REFERENCES:([t].[id_cliente]) OPTIMIZED)	Nested Loops	Inner Join	1	0	0,000001	304,1895
|--Nested Loops(Inner Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Inner Join	1	0	0,000001	304,1862
| |--Nested Loops(Inner Join, OUTER REFERENCES:([t].[id_persona]))	Nested Loops	Inner Join	1	0	0,000001	304,183
| | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([t].[id_ticket]))	Hash Match	Inner Join	1	0	15,942640	304,1797
| | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	783481	0	15,806310	252,7342
| | | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([i].[id_ingreso]))	Hash Match	Inner Join	783481	0	15,806310	252,7342
| | | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1027]))	Bitmap	Bitmap Create	783481	0	1,241181	211,8229
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	783481	0	1,241181	211,8229
| | | | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso], [l].[id_impuesto])=([li].[id_ingreso], [li].[id_impuesto]), RESIDUAL:([open10].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso]=[open10].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso] AND [open10].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]=[open10].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto]))	Hash Match	Inner Join	783481	0	22,225820	210,5817
| | | | | | |--Bitmap(HASH:([pg].[id_ingreso], [l].[id_impuesto]), DEFINE:([Bitmap1026]))	Bitmap	Bitmap Create	675916	0	1,117021	148,6098
| | | | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso], [l].[id_impuesto]))	Parallelism	Repartition Streams	675916	0	1,117021	148,6098
| | | | | | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	675916	0	1,963673	147,4928
| | | | | | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Opt_Bitmap1016]))	Bitmap	Bitmap Create	225311	0	0,206228	30,79273
| | | | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	225311	0	0,206228	30,79273
| | | | | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[pagos].[pagos_id_pagos] AS [pg]), WHERE:([open10].[dbo].[pagos].[total] as [pg].[total]>=(500) AND [open10].[dbo].[pagos].[total] as [pg].[total]<=(800)))	Clustered Index Scan	Clustered Index Scan	225311	25,63646	2,750039	28,3865
| | | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	675916	0	0,946437	114,7364
| | | | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1016],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([li].[id_ingreso], [li].[id_impuesto]))	Parallelism	Repartition Streams	10000000	0	10,440380	39,74613
| | | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [li]), WHERE:(PROBE([Bitmap1026],[open10].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso],[open10].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto])))	Clustered Index Scan	Clustered Index Scan	10000000	26,55572	2,750039	29,30576
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	10000000	0	8,606625	25,10497
| | | | |--Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_fecha] AS [i]), WHERE:(PROBE([Bitmap1027],[open10].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso],N'[IN ROW]')))	Index Scan	Index Scan	10000000	13,74831	2,750039	16,49835
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	9,814125	35,50285
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1028],[open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
| | |--Clustered Index Seek(OBJECT:([open10].[dbo].[persona].[persona_id_persona] AS [per]), SEEK:([per].[id_persona]=[open10].[dbo].[tickets].[id_persona] as [t].[id_persona]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
| |--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
|--Clustered Index Seek(OBJECT:([open10].[dbo].[clientes].[clientes_id_cliente] AS [cl]), SEEK:([cl].[id_cliente]=[open10].[dbo].[tickets].[id_cliente] as [t].[id_cliente]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831

ANEXO 82

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 4 SELECT USANDO ÍNDICE 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,028517	273,6663
|--Nested Loops(Inner Join, OUTER REFERENCES:([t].[id_cliente]) OPTIMIZED)	Nested Loops	Inner Join	1	0	0,000001	273,6378
|--Nested Loops(Inner Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Inner Join	1	0	0,000001	273,6345
| |--Nested Loops(Inner Join, OUTER REFERENCES:([t].[id_persona]))	Nested Loops	Inner Join	1	0	0,000001	273,6313
| | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([t].[id_ticket]))	Hash Match	Inner Join	1	0	15,867940	273,628
| | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	770439	0	15,733880	222,2572
| | | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([i].[id_ingreso]))	Hash Match	Inner Join	770439	0	15,733880	222,2572
| | | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1027]))	Bitmap	Bitmap Create	770439	0	1,220995	181,4183
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	770439	0	1,220995	181,4183
| | | | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso], [l].[id_impuesto])=([li].[id_ingreso], [li].[id_impuesto]), RESIDUAL:([open10].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso]=[open10].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso] AND [open10].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]=[open10].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto]))	Hash Match	Inner Join	770439	0	22,128130	180,1973
| | | | | | |--Bitmap(HASH:([pg].[id_ingreso], [l].[id_impuesto]), DEFINE:([Bitmap1026]))	Bitmap	Bitmap Create	664664	0	1,098900	118,3231
| | | | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso], [l].[id_impuesto]))	Parallelism	Repartition Streams	664664	0	1,098900	118,3231
| | | | | | | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([ll].[id_ticket]))	Hash Match	Inner Join	664664	0	1,931278	117,2242
| | | | | | | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Opt_Bitmap1016]))	Bitmap	Bitmap Create	221560	0	0,203269	0,5718071
| | | | | | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	221560	0	0,203269	0,5718071
| | | | | | | | | | |--Index Seek(OBJECT:([open10].[dbo].[pagos].[pagos_total] AS [pg]), SEEK:([pg].[total] >= (500) AND [pg].[total] <= (800)) ORDERED FORWARD)	Index Seek	Index Seek	221560	0,3075694	0,060968	0,3685377
| | | | | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([ll].[id_ticket]))	Parallelism	Repartition Streams	664664	0	0,931155	114,7211
| | | | | | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [ll]), WHERE:(PROBE([Opt_Bitmap1016],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
| | | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([li].[id_ingreso], [li].[id_impuesto]))	Parallelism	Repartition Streams	10000000	0	10,440380	39,74613
| | | | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [li]), WHERE:(PROBE([Bitmap1026],[open10].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso],[open10].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto])))	Clustered Index Scan	Clustered Index Scan	10000000	26,55572	2,750039	29,30576
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	10000000	0	8,606625	25,10497
| | | | | |--Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_fecha] AS [i]), WHERE:(PROBE([Bitmap1027],[open10].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso],N'[IN ROW]')))	Index Scan	Index Scan	10000000	13,74831	2,750039	16,49835
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	9,814125	35,50285
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1028],[open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
| | |--Clustered Index Seek(OBJECT:([open10].[dbo].[persona].[persona_id_persona] AS [per]), SEEK:([per].[id_persona]=[open10].[dbo].[tickets].[id_persona] as [t].[id_persona]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
| |--Clustered Index Seek(OBJECT:([open10].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open10].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
|--Clustered Index Seek(OBJECT:([open10].[dbo].[clientes].[clientes_id_cliente] AS [cl]), SEEK:([cl].[id_cliente]=[open10].[dbo].[tickets].[id_cliente] as [t].[id_cliente]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831

ANEXO 83

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 4 SELECT USANDO SUBQUERY 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1019]=[Expr1025], [Expr1023]=[open10].[dbo].[persona].[nombre] as [per].[nombre]))	Compute Scalar	Compute Scalar	675916	0	0,067592	994,7293
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	675916	0	0,028517	994,6617
|--Hash Match(Right Outer Join, HASH:([per].[id_persona])=([t].[id_persona]))	Hash Match	Right Outer Join	675916	0	1,460302	994,6332
|--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	10	0	0,028544	0,03183658
| |--Index Scan(OBJECT:([open10].[dbo].[persona].[persona_nombre] AS [per]))	Index Scan	Index Scan	10	0,003125	0,000168	0,003293
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([i].[id_ingreso], [l].[id_impuesto]))	Nested Loops	Left Outer Join	675916	0	0,706333	993,1411
|--Compute Scalar(DEFINE:([Expr1015]=[open10].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	675916	0	0,016898	289,6689
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso], [l].[id_impuesto]))	Parallelism	Repartition Streams	675916	0	2,830976	289,652
| |--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	675916	0	1,981808	286,821
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| | |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
| |--Compute Scalar(DEFINE:([Expr1011]=[open10].[dbo].[clientes].[nombres] as [cl].[nombres]))	Compute Scalar	Compute Scalar	675916	0	0,016898	284,3899
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	675916	0	2,228988	284,373
| |--Hash Match(Left Outer Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]=[open10].[dbo].[tickets].[id_cliente] as [t].[id_cliente]))	Hash Match	Left Outer Join	675916	0	11,317560	282,144
| |--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1036]))	Bitmap	Bitmap Create	675916	0	1,646010	237,1109
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	675916	0	1,646010	237,1109
| | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([t].[id_ticket]))	Hash Match	Inner Join	675916	0	12,391900	235,4649
| | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1035]))	Bitmap	Bitmap Create	67592	0	0,081110	187,5701
| | | |--Filter(WHERE:([open10].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso]=[open10].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso]))	Filter	Filter	67592	0	0,081110	187,5701
| | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([t].[id_ticket]))	Hash Match	Inner Join	675916	0	1,963673	187,489
| | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Opt_Bitmap1027]))	Bitmap	Bitmap Create	225311	0	0,206228	30,79273
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	225311	0	0,206228	30,79273
| | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[pagos].[pagos_id_pagos] AS [pg]), WHERE:([open10].[dbo].[pagos].[total] as [pg].[total]>=(500) AND [open10].[dbo].[pagos].[total] as [pg].[total]<=(800)))	Clustered Index Scan	Clustered Index Scan	225311	25,63646	2,750039	28,3865
| | | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([i].[id_ingreso]))	Hash Match	Inner Join	675916	0	14,891270	154,7326
| | | | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	675916	0	0,946437	114,7364
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	675916	0	0,946437	114,7364
| | | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1027],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	10000000	0	8,606625	25,10497
| | | | |--Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_fecha] AS [i]), WHERE:(PROBE([Bitmap1034],[open10].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso],N'[IN ROW]')))	Index Scan	Index Scan	10000000	13,74831	2,750039	16,49835
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	9,814125	35,50285
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1035],[open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	3,728791	33,71559
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1036],[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,37053	0,616264	29,9868
|--Index Spool(SEEK:([i].[id_ingreso]=[open10].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso] AND [l].[id_impuesto]=[open10].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]))	Index Spool	Lazy Spool	1	0,003125	0,000258	702,7658
|--Assert(WHERE:(CASE WHEN [Expr1024]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	525,7209
|--Stream Aggregate(DEFINE:([Expr1024]=Count(*), [Expr1025]=ANY([open10].[dbo].[lineasimpuesto].[monto] as [li].[monto])))	Stream Aggregate	Aggregate	1	0	0,000001	525,6127
|--Index Spool(SEEK:([li].[id_ingreso]=[open10].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso] AND [li].[id_impuesto]=[open10].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]))	Index Spool	Eager Spool	1	226,7496	10,000260	525,3391
|--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [li]))	Clustered Index Scan	Clustered Index Scan	10000000	26,55572	11,000160	37,55587

ANEXO 84

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 4 SELECT USANDO JOIN 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	626299	0	2,633826	343,6653
|--Hash Match(Inner Join, HASH:([pg].[id_ingreso], [l].[id_impuesto])=([li].[id_ingreso], [li].[id_impuesto]), RESIDUAL:([open10].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso]=[open10].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso] AND [open10].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]=[open10].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto]))	Hash Match	Inner Join	626299	0	17,2468	341,0314
|--Bitmap(HASH:([pg].[id_ingreso], [l].[id_impuesto]), DEFINE:([Bitmap1030]))	Bitmap	Bitmap Create	6759	0	0,05920646	284,0385
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso], [l].[id_impuesto]))	Parallelism	Repartition Streams	6759	0	0,05920646	284,0385
|--Hash Match(Inner Join, HASH:([l].[id_producto])=([p].[id_producto]))	Hash Match	Inner Join	6759	0	1,167273	283,9793
|--Bitmap(HASH:([l].[id_producto]), DEFINE:([Bitmap1029]))	Bitmap	Bitmap Create	67592	0	0,2753657	282,3627
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	67592	0	0,2753657	282,3627
|--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([i].[id_ingreso]))	Hash Match	Inner Join	67592	0	12,49775	282,0873
|--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	67592	0	0,08110996	244,4846
|--Filter(WHERE:([open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket]=[open10].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso]))	Filter	Filter	67592	0	0,08110996	244,4846
|--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	675916	0	1,963673	244,4035
|--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Opt_Bitmap1024]))	Bitmap	Bitmap Create	225311	0	0,2062281	30,79273
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	225311	0	0,2062281	30,79273
|--Clustered Index Scan(OBJECT:([open10].[dbo].[pagos].[pagos_id_pagos] AS [pg]), WHERE:([open10].[dbo].[pagos].[total] as [pg].[total]>=(500) AND [open10].[dbo].[pagos].[total] as [pg].[total]<=(800)))	Clustered Index Scan	Clustered Index Scan	225311	25,63646	2,750039	28,3865
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	675916	0	0,02851275	211,6471
|--Hash Match(Inner Join, HASH:([per].[id_persona])=([t].[id_persona]))	Hash Match	Inner Join	675916	0	1,460302	211,6186
|--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	10	0	0,02854357	0,03183658
|--Index Scan(OBJECT:([open10].[dbo].[persona].[persona_nombre] AS [per]))	Index Scan	Index Scan	10	0,003125	0,000168	0,003293
|--Hash Match(Inner Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open10].[dbo].[tickets].[id_cliente] as [t].[id_cliente]=[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]))	Hash Match	Inner Join	675916	0	9,949176	210,1264
|--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1027]))	Bitmap	Bitmap Create	675916	0	1,331202	166,4617
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	675916	0	1,331202	166,4617
|--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	675916	0	14,89127	165,1305
|--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1026]))	Bitmap	Bitmap Create	675916	0	0,9464365	114,7364
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	675916	0	0,9464365	114,7364
|--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1024],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	9,814125	35,50285
|--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1026],[open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	3,728791	33,71559
|--Clustered Index Scan(OBJECT:([open10].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1027],[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,37053	0,6162642	29,9868
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	10000000	0	8,606625	25,10497
|--Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_fecha] AS [i]), WHERE:(PROBE([Bitmap1028],[open10].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso],N'[IN ROW]')))	Index Scan	Index Scan	10000000	13,74831	2,750039	16,49835
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,1585623	0,4493123
|--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]), WHERE:(PROBE([Bitmap1029],[open10].[dbo].[productos].[id_producto] as [p].[id_producto],N'[IN ROW]')))	Index Scan	Index Scan	78763	0,2690509	0,02169907	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([li].[id_ingreso], [li].[id_impuesto]))	Parallelism	Repartition Streams	10000000	0	10,44038	39,74613
|--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [li]), WHERE:(PROBE([Bitmap1030],[open10].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso],[open10].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto])))	Clustered Index Scan	Clustered Index Scan	10000000	26,55572	2,750039	29,30576

ANEXO 85

PLAN DEVEJECUCIÓN EN SQL SERVER 2008 CONSULTA 4 SELECT USANDO SUBQUERY + JOIN 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1021]=[open10].[dbo].[persona].[nombre] as [per].[nombre]))	Compute Scalar	Compute Scalar	783467	0	0,07834674	348,5608
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	783467	0	0,02851664	348,4825
|--Hash Match(Right Outer Join, HASH:([per].[id_persona])=([t].[id_persona]))	Hash Match	Right Outer Join	783467	0	1,689831	348,4539
|--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	10	0	0,02854357	0,03183658
| |--Index Scan(OBJECT:([open10].[dbo].[persona].[persona_nombre] AS [per]))	Index Scan	Index Scan	10	0,003125	0,000168	0,003293
|--Compute Scalar(DEFINE:([Expr1017]=[open10].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	783467	0	0,01958669	346,7323
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	783467	0	2,187204	346,7127
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,1585623	0,4493123
| |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,02169907	0,29075
|--Compute Scalar(DEFINE:([Expr1013]=[open10].[dbo].[clientes].[nombres] as [cl].[nombres]))	Compute Scalar	Compute Scalar	783467	0	0,01958669	344,0762
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([ll].[id_producto]))	Parallelism	Repartition Streams	783467	0	2,430464	344,0566
|--Hash Match(Left Outer Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]=[open10].[dbo].[tickets].[id_cliente] as [t].[id_cliente]))	Hash Match	Left Outer Join	783467	0	12,00641	341,6261
|--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1031]))	Bitmap	Bitmap Create	783467	0	1,754724	295,9041
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	783467	0	1,754724	295,9041
| |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([li].[id_ingreso]))	Hash Match	Inner Join	783467	0	12,54104	294,1494
| |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1030]))	Bitmap	Bitmap Create	78363	0	0,1957502	256,5034
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	78363	0	0,1957502	256,5034
| | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso], [l].[id_impuesto])=([li].[id_ingreso], [li].[id_impuesto]), RESIDUAL:([open10].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso]=[open10].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso] AND [open10].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]=[open10].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto]))	Hash Match	Inner Join	78363	0	17,08008	256,3076
| | |--Bitmap(HASH:([pg].[id_ingreso], [l].[id_impuesto]), DEFINE:([Bitmap1029]))	Bitmap	Bitmap Create	67592	0	0,08110996	199,4814
| | | |--Filter(WHERE:([open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket]=[open10].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso]))	Filter	Filter	67592	0	0,08110996	199,4814
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso], [l].[id_impuesto]))	Parallelism	Repartition Streams	675916	0	1,513446	199,4003
| | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	675916	0	1,963673	197,8869
| | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Opt_Bitmap1026]))	Bitmap	Bitmap Create	225311	0	0,2062281	30,79273
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	225311	0	0,2062281	30,79273
| | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[pagos].[pagos_id_pagos] AS [pg]), WHERE:([open10].[dbo].[pagos].[total] as [pg].[total]>=(500) AND [open10].[dbo].[pagos].[total] as [pg].[total]<=(800)))	Clustered Index Scan	Clustered Index Scan	225311	25,63646	2,750039	28,3865
| | | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	675916	0	14,89127	165,1305
| | | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	675916	0	0,9464365	114,7364
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	675916	0	0,9464365	114,7364
| | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1026],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	9,814125	35,50285
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1028],[open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([li].[id_ingreso], [li].[id_impuesto]))	Parallelism	Repartition Streams	10000000	0	10,44038	39,74613
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [li]), WHERE:(PROBE([Bitmap1029],[open10].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso],[open10].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto])))	Clustered Index Scan	Clustered Index Scan	10000000	26,55572	2,750039	29,30576
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	10000000	0	8,606625	25,10497
| |--Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_fecha] AS [i]), WHERE:(PROBE([Bitmap1030],[open10].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso],N'[IN ROW]')))	Index Scan	Index Scan	10000000	13,74831	2,750039	16,49835
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	3,728791	33,71559
|--Clustered Index Scan(OBJECT:([open10].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1031],[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,37053	0,6162642	29,9868

ANEXO 86

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 4 SELECT USANDO ÍNDICE + JOIN 10 MILLONES DE DATOS..

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	615874	0	2,590457	313,0902
|--Hash Match(Inner Join, HASH:([pg].[id_ingreso], [l].[id_impuesto])=([li].[id_ingreso], [li].[id_impuesto]), RESIDUAL:([open10].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso]=[open10].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso] AND [open10].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]=[open10].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto]))	Hash Match	Inner Join	615874	0	17,23199	310,4998
|--Bitmap(HASH:([pg].[id_ingreso], [l].[id_impuesto]), DEFINE:([Bitmap1030]))	Bitmap	Bitmap Create	6647	0	0,05869528	253,5217
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso], [l].[id_impuesto]))	Parallelism	Repartition Streams	6647	0	0,05869528	253,5217
| |--Hash Match(Inner Join, HASH:([l].[id_producto])=([p].[id_producto]))	Hash Match	Inner Join	6647	0	1,149636	253,463
| |--Bitmap(HASH:([l].[id_producto]), DEFINE:([Bitmap1029]))	Bitmap	Bitmap Create	66466	0	0,2712561	251,864
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	66466	0	0,2712561	251,864
| | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([i].[id_ingreso]))	Hash Match	Inner Join	66466	0	12,48039	251,5927
| | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	66466	0	0,07975969	214,0074
| | | |--Filter(WHERE:([open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket]=[open10].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso]))	Filter	Filter	66466	0	0,07975969	214,0074
| | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([ll].[id_ticket]))	Hash Match	Inner Join	664664	0	1,931278	213,9276
| | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Opt_Bitmap1024]))	Bitmap	Bitmap Create	221560	0	0,2032694	0,5718071
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	221560	0	0,2032694	0,5718071
| | | | |--Index Seek(OBJECT:([open10].[dbo].[pagos].[pagos_total] AS [pg]), SEEK:([pg].[total] >= (500) AND [pg].[total] <= (800)) ORDERED FORWARD)	Index Seek	Index Seek	221560	0,3075694	0,06096828	0,3685377
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([ll].[id_ticket]))	Parallelism	Repartition Streams	664664	0	0,02851275	211,4245
| | | |--Hash Match(Inner Join, HASH:([per].[id_persona])=([t].[id_persona]))	Hash Match	Inner Join	664664	0	1,436289	211,396
| | | |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	10	0	0,02854357	0,03183658
| | | | |--Index Scan(OBJECT:([open10].[dbo].[persona].[persona_nombre] AS [per]))	Index Scan	Index Scan	10	0,003125	0,000168	0,003293
| | | |--Hash Match(Inner Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open10].[dbo].[tickets].[id_cliente] as [t].[id_cliente]=[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]))	Hash Match	Inner Join	664664	0	9,844795	209,9279
| | | |--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1027]))	Bitmap	Bitmap Create	664664	0	1,309515	166,3675
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	664664	0	1,309515	166,3675
| | | | |--Hash Match(Inner Join, HASH:([ll].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	664664	0	14,83407	165,058
| | | | |--Bitmap(HASH:([ll].[id_ticket]), DEFINE:([Bitmap1026]))	Bitmap	Bitmap Create	664664	0	0,9311554	114,7211
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([ll].[id_ticket]))	Parallelism	Repartition Streams	664664	0	0,9311554	114,7211
| | | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [ll]), WHERE:(PROBE([Opt_Bitmap1024],[open10].[dbo].[lineasticket].[id_ticket] as [ll].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	9,814125	35,50285
| | | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1026],[open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	3,728791	33,71559
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1027],[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,37053	0,6162642	29,9868
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	10000000	0	8,606625	25,10497
| | |--Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_fecha] AS [i]), WHERE:(PROBE([Bitmap1028],[open10].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso],N'[IN ROW]')))	Index Scan	Index Scan	10000000	13,74831	2,750039	16,49835
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,1585623	0,4493123
| |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]), WHERE:(PROBE([Bitmap1029],[open10].[dbo].[productos].[id_producto] as [p].[id_producto],N'[IN ROW]')))	Index Scan	Index Scan	78763	0,2690509	0,02169907	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([li].[id_ingreso], [li].[id_impuesto]))	Parallelism	Repartition Streams	10000000	0	10,44038	39,74613
|--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [li]), WHERE:(PROBE([Bitmap1030],[open10].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso],[open10].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto])))	Clustered Index Scan	Clustered Index Scan	10000000	26,55572	2,750039	29,30576

ANEXO 87

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 4 SELECT CON ÍNDICE + SUBQUERY 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1019]=[Expr1025], [Expr1023]=[open10].[dbo].[persona].[nombre] as [per].[nombre]))	Compute Scalar	Compute Scalar	664664	0	0,066466	957,4116
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	664664	0	0,028517	957,3451
|--Hash Match(Right Outer Join, HASH:([per].[id_persona])=([t].[id_persona]))	Hash Match	Right Outer Join	664664	0	1,436289	957,3166
|--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	10	0	0,028544	0,03183658
| |--Index Scan(OBJECT:([open10].[dbo].[persona].[persona_nombre] AS [per]))	Index Scan	Index Scan	10	0,003125	0,000168	0,003293
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([i].[id_ingreso], [l].[id_impuesto]))	Nested Loops	Left Outer Join	664664	0	0,694574	955,8484
|--Compute Scalar(DEFINE:([Expr1015]=[open10].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	664664	0	0,016617	259,0667
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso], [l].[id_impuesto]))	Parallelism	Repartition Streams	664664	0	2,784322	259,0501
| |--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	664664	0	1,960319	256,2658
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| | |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
| |--Compute Scalar(DEFINE:([Expr1011]=[open10].[dbo].[clientes].[nombres] as [cl].[nombres]))	Compute Scalar	Compute Scalar	664664	0	0,016617	253,8562
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	664664	0	2,192355	253,8395
| |--Hash Match(Left Outer Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]=[open10].[dbo].[tickets].[id_cliente] as [t].[id_cliente]))	Hash Match	Left Outer Join	664664	0	11,190400	251,6472
| |--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1036]))	Bitmap	Bitmap Create	664664	0	1,619083	206,7412
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	664664	0	1,619083	206,7412
| | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([t].[id_ticket]))	Hash Match	Inner Join	664664	0	12,376310	205,1221
| | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1035]))	Bitmap	Bitmap Create	66466	0	0,079760	157,243
| | | |--Filter(WHERE:([open10].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso]=[open10].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso]))	Filter	Filter	66466	0	0,079760	157,243
| | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([ll].[id_ticket]))	Hash Match	Inner Join	664664	0	1,931278	157,1632
| | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Opt_Bitmap1027]))	Bitmap	Bitmap Create	221560	0	0,203269	0,5718071
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	221560	0	0,203269	0,5718071
| | | |--Index Seek(OBJECT:([open10].[dbo].[pagos].[pagos_total] AS [pg]), SEEK:([pg].[total] >= (500) AND [pg].[total] <= (800)) ORDERED FORWARD)	Index Seek	Index Seek	221560	0,3075694	0,060968	0,3685377
| | |--Hash Match(Inner Join, HASH:([ll].[id_ticket])=([i].[id_ingreso]))	Hash Match	Inner Join	664664	0	14,834070	154,6601
| | |--Bitmap(HASH:([ll].[id_ticket]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	664664	0	0,931155	114,7211
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([ll].[id_ticket]))	Parallelism	Repartition Streams	664664	0	0,931155	114,7211
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1027],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	10000000	0	8,606625	25,10497
| | |--Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_fecha] AS [i]), WHERE:(PROBE([Bitmap1034],[open10].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso],N'[IN ROW]')))	Index Scan	Index Scan	10000000	13,74831	2,750039	16,49835
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	9,814125	35,50285
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1035],[open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	3,728791	33,71559
|--Clustered Index Scan(OBJECT:([open10].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1036],[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,37053	0,616264	29,9868
|--Index Spool(SEEK:([i].[id_ingreso]=[open10].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso] AND [l].[id_impuesto]=[open10].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]))	Index Spool	Lazy Spool	1	0,003125	0,000258	696,0872
|--Assert(WHERE:(CASE WHEN [Expr1024]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	521,9908
|--Stream Aggregate(DEFINE:([Expr1024]=Count(*), [Expr1025]=ANY([open10].[dbo].[lineasimpuesto].[monto] as [li].[monto])))	Stream Aggregate	Aggregate	1	0	0,000001	521,8844
|--Index Spool(SEEK:([li].[id_ingreso]=[open10].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso] AND [li].[id_impuesto]=[open10].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]))	Index Spool	Eager Spool	1	226,7496	10,000260	521,6152
|--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [li]))	Clustered Index Scan	Clustered Index Scan	10000000	26,55572	11,000160	37,55587

ANEXO 88

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 4 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 10 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1021]=[open10].[dbo].[persona].[nombre] as [per].[nombre]))	Compute Scalar	Compute Scalar	770426	0	0,07704256	317,914
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	770426	0	0,02851664	317,837
|--Hash Match(Right Outer Join, HASH:([per].[id_persona])=([t].[id_persona]))	Hash Match	Right Outer Join	770426	0	1,661998	317,8085
|--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	10	0	0,02854357	0,03183658
| |--Index Scan(OBJECT:([open10].[dbo].[persona].[persona_nombre] AS [per]))	Index Scan	Index Scan	10	0,003125	0,000168	0,003293
|--Compute Scalar(DEFINE:([Expr1017]=[open10].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	770426	0	0,01926064	316,1146
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	770426	0	2,162297	316,0954
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,1585623	0,4493123
| |--Index Scan(OBJECT:([open10].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,02169907	0,29075
|--Compute Scalar(DEFINE:([Expr1013]=[open10].[dbo].[clientes].[nombres] as [cl].[nombres]))	Compute Scalar	Compute Scalar	770426	0	0,01926064	313,4838
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	770426	0	2,390481	313,4645
|--Hash Match(Left Outer Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]=[open10].[dbo].[tickets].[id_cliente] as [t].[id_cliente]))	Hash Match	Left Outer Join	770426	0	11,86779	311,074
|--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1031]))	Bitmap	Bitmap Create	770426	0	1,725988	265,4906
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	770426	0	1,725988	265,4906
| |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([i].[id_ingreso]))	Hash Match	Inner Join	770426	0	12,52297	263,7646
| |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1030]))	Bitmap	Bitmap Create	77058	0	0,1929666	226,1367
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	77058	0	0,1929666	226,1367
| | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso], [l].[id_impuesto])=([ii].[id_ingreso], [li].[id_impuesto]), RESIDUAL:([open10].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso]=[open10].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso] AND [open10].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]=[open10].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto]))	Hash Match	Inner Join	77058	0	17,06804	225,9437
| | |--Bitmap(HASH:([pg].[id_ingreso], [l].[id_impuesto]), DEFINE:([Bitmap1029]))	Bitmap	Bitmap Create	66466	0	0,07975969	169,1296
| | | |--Filter(WHERE:([open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket]=[open10].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso]))	Filter	Filter	66466	0	0,07975969	169,1296
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso], [l].[id_impuesto]))	Parallelism	Repartition Streams	664664	0	1,488725	169,0498
| | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	664664	0	1,931278	167,5611
| | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Opt_Bitmap1026]))	Bitmap	Bitmap Create	221560	0	0,2032694	0,5718071
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	221560	0	0,2032694	0,5718071
| | | | |--Index Seek(OBJECT:([open10].[dbo].[pagos].[pagos_total] AS [pg]), SEEK:([pg].[total] >= (500) AND [pg].[total] <= (800)) ORDERED FORWARD)	Index Seek	Index Seek	221560	0,3075694	0,06096828	0,3685377
| | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	664664	0	14,83407	165,058
| | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	664664	0	0,9311554	114,7211
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	664664	0	0,9311554	114,7211
| | | |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1026],[open10].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	29999270	101,9402	8,249838	110,19
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	10000000	0	9,814125	35,50285
| | |--Clustered Index Scan(OBJECT:([open10].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1028],[open10].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	10000000	22,93868	2,750039	25,68872
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([li].[id_ingreso], [li].[id_impuesto]))	Parallelism	Repartition Streams	10000000	0	10,44038	39,74613
| |--Clustered Index Scan(OBJECT:([open10].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [li]), WHERE:(PROBE([Bitmap1029],[open10].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso],[open10].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto])))	Clustered Index Scan	Clustered Index Scan	10000000	26,55572	2,750039	29,30576
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	10000000	0	8,606625	25,10497
| |--Index Scan(OBJECT:([open10].[dbo].[ingresos].[ingresos_fecha] AS [i]), WHERE:(PROBE([Bitmap1030],[open10].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso],N'[IN ROW]')))	Index Scan	Index Scan	10000000	13,74831	2,750039	16,49835
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	3,728791	33,71559
|--Clustered Index Scan(OBJECT:([open10].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1031],[open10].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,37053	0,6162642	29,9868

ANEXO 89

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 1 SELECT SIMPLE y SELECT USANDO JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1905849	0	7,233324	544,4504
|--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	1905849	0	3,801629	537,2171
|--Bitmap(HASH:([p].[id_producto]), DEFINE:([Opt_Bitmap1026]))	Bitmap	Bitmap Create	78763	0	0,158562	0,4493123
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open20].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	1424901	0	4,126605	532,9661
|--Hash Match(Inner Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open20].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]=[open20].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Inner Join	1424901	0	3,923890	528,8395
|--Bitmap(HASH:([a].[id_atributoconjuntoinstancia]), DEFINE:([Opt_Bitmap1024]))	Bitmap	Bitmap Create	1106	0	0,030326	0,03897994
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,030326	0,03897994
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,000343	0,008653585
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1424901	0	2,627253	524,8766
|--Hash Match(Inner Join, HASH:([t].[id_ticket])=([l].[id_ticket]))	Hash Match	Inner Join	1424901	0	19,025700	522,2494
|--Bitmap(HASH:([t].[id_ticket]), DEFINE:([Opt_Bitmap1023]))	Bitmap	Bitmap Create	6666667	0	4,022250	87,85477
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	6666667	0	4,022250	87,85477
| |--Hash Match(Inner Join, HASH:([ps].[id_persona])=([t].[id_persona]))	Hash Match	Inner Join	6666667	0	32,425400	83,83253
| |--Bitmap(HASH:([ps].[id_persona]), DEFINE:([Bitmap1032]))	Bitmap	Bitmap Create	3	0	0,028506	0,03655184
| | |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	3	0	0,028506	0,03655184
| | |--Nested Loops(Inner Join, OUTER REFERENCES:([ps].[id_rol]))	Nested Loops	Inner Join	3	0	0,000042	0,0080456
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[persona].[persona_id_persona] AS [ps]))	Clustered Index Scan	Clustered Index Scan	10	0,003125	0,000168	0,003293
| | |--Clustered Index Seek(OBJECT:([open20].[dbo].[roles].[roles_id_rol] AS [r]), SEEK:([r].[id_rol]=[open20].[dbo].[persona].[id_rol] as [ps].[id_rol]), WHERE:([open20].[dbo].[roles].[nombre] as [r].[nombre]='Dependiente') ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,004706
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1032],[open20].[dbo].[tickets].[id_persona] as [t].[id_persona],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,500039	51,37057
|--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	1424901	0	6,851585	415,3689
|--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1022]))	Bitmap	Bitmap Create	1687233	0	1,039258	56,81057
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	1687233	0	1,039258	56,81057
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[ingresos].[ingresos_id_ingreso] AS [i]), WHERE:([open20].[dbo].[ingresos].[fecha] as [i].[fecha]>='2013-01-01 00:00:00.000' AND [open20].[dbo].[ingresos].[fecha] as [i].[fecha]<='2013-01-31 00:00:00.000'))	Clustered Index Scan	Clustered Index Scan	1687233	45,87127	5,500039	51,37131
|--Hash Match(Inner Join, HASH:([l].[id_ticket])=([im].[id_ingreso]))	Hash Match	Inner Join	1424901	0	30,583190	351,7068
|--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1033]))	Bitmap	Bitmap Create	1261369	0	0,031534	244,3367
| |--Compute Scalar(DEFINE:([Expr1016]=CONVERT_IMPLICIT(numeric(10,0),[open20].[dbo].[lineasticket].[unidades] as [l].[unidades]*[open20].[dbo].[lineasticket].[precio] as [l].[precio],0)*(0.12)))	Compute Scalar	Compute Scalar	1261369	0	0,031534	244,3367
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	1261369	0	1,741517	244,3052
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1022],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1023],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1024],[open20].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia],N'[IN ROW]') AND PROBE([Opt_Bitmap1026],[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,498980	220,3651
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([im].[id_ingreso]))	Parallelism	Repartition Streams	19971430	0	15,782220	76,78686
|--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [im]), WHERE:([open20].[dbo].[lineasimpuesto].[monto] as [im].[monto]<(500) AND PROBE([Bitmap1033],[open20].[dbo].[lineasimpuesto].[id_ingreso] as [im].[id_ingreso],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	19971430	53,10461	5,500039	58,60464

ANEXO 90

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 1 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1883081	0	7,147252	398,1222
|--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	1883081	0	3,764376	390,9749
|--Bitmap(HASH:([p].[id_producto]), DEFINE:([Opt_Bitmap1026]))	Bitmap	Bitmap Create	78763	0	0,158562	0,4493123
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open20].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	1407795	0	4,077406	386,7612
|--Hash Match(Inner Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open20].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]=[open20].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Inner Join	1407795	0	3,877158	382,6838
|--Bitmap(HASH:([a].[id_atributoconjuntoinstancia]), DEFINE:([Opt_Bitmap1024]))	Bitmap	Bitmap Create	1106	0	0,030326	0,03897994
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,030326	0,03897994
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,000343	0,008653585
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1407795	0	2,596054	378,7677
|--Hash Match(Inner Join, HASH:([t].[id_ticket])=([l].[id_ticket]))	Hash Match	Inner Join	1407795	0	18,992590	376,1716
|--Bitmap(HASH:([t].[id_ticket]), DEFINE:([Opt_Bitmap1023]))	Bitmap	Bitmap Create	6666667	0	4,022250	23,06254
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	6666667	0	4,022250	23,06254
| |--Nested Loops(Inner Join, OUTER REFERENCES:([ps].[id_persona]) OPTIMIZED)	Nested Loops	Inner Join	6666667	0	6,966667	19,04029
| |--Nested Loops(Inner Join, OUTER REFERENCES:([ps].[id_rol]))	Nested Loops	Inner Join	3	0	0,000010	0,03654058
| | |--Parallelism(Distribute Streams, RoundRobin Partitioning)	Parallelism	Distribute Streams	10	0	0,028526	0,03181932
| | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[persona].[persona_id_persona] AS [ps]))	Clustered Index Scan	Clustered Index Scan	10	0,003125	0,000168	0,003293
| | |--Clustered Index Seek(OBJECT:([open20].[dbo].[roles].[roles_id_rol] AS [r]), SEEK:([r].[id_rol]=[open20].[dbo].[persona].[id_rol] as [ps].[id_rol]), WHERE:([open20].[dbo].[roles].[nombre] as [r].[nombre]='Dependiente') ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,004706
| |--Index Seek(OBJECT:([open20].[dbo].[tickets].[Ix_persona_tickets>] AS [t]), SEEK:([t].[id_persona]=[open20].[dbo].[persona].[id_persona] as [ps].[id_persona]) ORDERED FORWARD)	Index Seek	Index Seek	2000000	2,015718	2,200157	12,03708
|--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	1407795	0	6,769784	334,1165
|--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1022]))	Bitmap	Bitmap Create	1667076	0	1,027183	3,779904
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	1667076	0	1,027183	3,779904
| |--Index Seek(OBJECT:([open20].[dbo].[ingresos].[Ix_ingreso_fecha] AS [i]), SEEK:([i].[fecha] >= '2013-01-01 00:00:00.000' AND [i].[fecha] <= '2013-01-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	1667076	2,294236	0,458485	2,752721
|--Hash Match(Inner Join, HASH:([l].[id_ticket])=([im].[id_ingreso]))	Hash Match	Inner Join	1407795	0	30,501270	323,5668
|--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1032]))	Bitmap	Bitmap Create	1246225	0	0,031156	244,3158
| |--Compute Scalar(DEFINE:([Expr1016]=CONVERT_IMPLICIT(numeric(10,0),[open20].[dbo].[lineasticket].[unidades] as [l].[unidades]*[open20].[dbo].[lineasticket].[precio] as [l].[precio],0)*(0.12)))	Compute Scalar	Compute Scalar	1246225	0	0,031156	244,3158
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	1246225	0	1,720952	244,2846
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1022],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1023],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1024],[open20].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia],N'[IN ROW]') AND PROBE([Opt_Bitmap1026],[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,498980	220,3651
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([im].[id_ingreso]))	Parallelism	Repartition Streams	19980870	0	15,789660	48,74978
|--Index Seek(OBJECT:([open20].[dbo].[lineasimpuesto].[limpuesto_monto] AS [im]), SEEK:([im].[monto] < (500)), WHERE:(PROBE([Bitmap1032],[open20].[dbo].[lineasimpuesto].[id_ingreso] as [im].[id_ingreso],N'[IN ROW]')) ORDERED FORWARD)	Index Seek	Index Seek	19980870	27,46535	5,494778	32,96012

ANEXO 91

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 1 SELECT USANDO SUBQUERY 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1031]=[open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]))	Compute Scalar	Compute Scalar	5049684	0	0,504968	782,6179
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	5049684	0	11,446470	782,113
|--Hash Match(Right Outer Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open20].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]=[open20].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Right Outer Join	5049684	0	13,982830	770,6665
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,030326	0,03897994
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,000343	0,008653585
|--Compute Scalar(DEFINE:([Expr1027]=[open20].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	5049684	0	0,126242	756,6447
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	5049684	0	13,593530	756,5184
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	5049684	0	10,334610	742,9249
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open20].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	1401257	0	1,745807	732,141
|--Hash Match(Inner Join, HASH:([open20].[dbo].[persona].[id_persona])=([open20].[dbo].[tickets].[id_persona]))	Hash Match	Inner Join	1401257	0	8,014923	730,3951
|--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	3	0	0,028505	80,31606
| |--Nested Loops(Left Semi Join, OUTER REFERENCES:([open20].[dbo].[persona].[id_persona]))	Nested Loops	Left Semi Join	3	0	0,000014	80,28755
| |--Parallelism(Gather Streams)	Parallelism	Gather Streams	3	0	0,028502	0,06504245
| | |--Nested Loops(Inner Join, OUTER REFERENCES:([open20].[dbo].[persona].[id_rol]))	Nested Loops	Inner Join	3	0	0,000010	0,03654058
| | |--Parallelism(Distribute Streams, RoundRobin Partitioning)	Parallelism	Distribute Streams	10	0	0,028526	0,03181932
| | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[persona].[persona_id_persona]))	Clustered Index Scan	Clustered Index Scan	10	0,003125	0,000168	0,003293
| | |--Clustered Index Seek(OBJECT:([open20].[dbo].[roles].[roles_id_rol]), SEEK:([open20].[dbo].[roles].[id_rol]=[open20].[dbo].[persona].[id_rol]), WHERE:([open20].[dbo].[roles].[nombre]='Dependiente') ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,004706
| |--Merge Join(Left Semi Join, MERGE:([open20].[dbo].[tickets].[id_ticket])=([open20].[dbo].[ingresos].[id_ingreso]), RESIDUAL:([open20].[dbo].[tickets].[id_ticket]=[open20].[dbo].[ingresos].[id_ingreso]))	Merge Join	Left Semi Join	28264	0	7,765661	80,2225
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket]), WHERE:([open20].[dbo].[persona].[id_persona]=[open20].[dbo].[tickets].[id_persona]) ORDERED FORWARD)	Clustered Index Scan	Clustered Index Scan	335036	45,87053	22,000160	30,31757
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[ingresos].[ingresos_id_ingreso]), WHERE:([open20].[dbo].[ingresos].[fecha]>='2013-01-01 00:00:00.000' AND [open20].[dbo].[ingresos].[fecha]<='2013-01-31 00:00:00.000') ORDERED FORWARD)	Clustered Index Scan	Clustered Index Scan	282642	45,87127	22,000160	30,31798
|--Hash Match(Inner Join, HASH:([open20].[dbo].[tickets].[id_ticket])=([l].[id_ticket]))	Hash Match	Inner Join	5049684	0	77,400720	642,0641
|--Bitmap(HASH:([open20].[dbo].[tickets].[id_ticket]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	1683336	0	28,285320	261,2914
| |--Hash Match(Inner Join, HASH:([open20].[dbo].[lineasimpuesto].[id_ingreso])=([open20].[dbo].[tickets].[id_ticket]))	Hash Match	Inner Join	1683336	0	28,285320	261,2914
| |--Bitmap(HASH:([open20].[dbo].[lineasimpuesto].[id_ingreso]), DEFINE:([Bitmap1033]))	Bitmap	Bitmap Create	1683336	0	7,590271	165,8308
| | |--Hash Match(Aggregate, HASH:([open20].[dbo].[lineasimpuesto].[id_ingreso]))	Hash Match	Aggregate	1683336	0	7,590271	165,8308
| | |--Hash Match(Inner Join, HASH:([open20].[dbo].[ingresos].[id_ingreso])=([open20].[dbo].[lineasimpuesto].[id_ingreso]))	Hash Match	Inner Join	1905971	0	28,432680	158,2405
| | |--Bitmap(HASH:([open20].[dbo].[ingresos].[id_ingreso]), DEFINE:([Bitmap1032]))	Bitmap	Bitmap Create	1687233	0	1,039258	56,81057
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open20].[dbo].[ingresos].[id_ingreso]))	Parallelism	Repartition Streams	1687233	0	1,039258	56,81057
| | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[ingresos].[ingresos_id_ingreso]), WHERE:([open20].[dbo].[ingresos].[fecha]>='2013-01-01 00:00:00.000' AND [open20].[dbo].[ingresos].[fecha]<='2013-01-31 00:00:00.000'))	Clustered Index Scan	Clustered Index Scan	1687233	45,87127	5,500039	51,37131
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open20].[dbo].[lineasimpuesto].[id_ingreso]))	Parallelism	Repartition Streams	19971430	0	11,992640	72,99728
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos]), WHERE:([open20].[dbo].[lineasimpuesto].[monto]<(500) AND PROBE([Bitmap1032],[open20].[dbo].[lineasimpuesto].[id_ingreso],N'[IN ROW]'))	Clustered Index Scan	Clustered Index Scan	19971430	53,10461	5,500039	58,60464
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open20].[dbo].[tickets].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	15,804750	67,17532
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket]), WHERE:(PROBE([Bitmap1033],[open20].[dbo].[tickets].[id_ticket],N'[IN ROW]'))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,500039	51,37057
|--Compute Scalar(DEFINE:([Expr1023]=CONVERT_IMPLICIT(numeric(10,0),[open20].[dbo].[lineasticket].[unidades] as [l].[unidades]*[open20].[dbo].[lineasticket].[precio] as [l].[precio],0)*(0.12)))	Compute Scalar	Compute Scalar	59996160	0	1,499904	303,372
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	59996160	0	81,507030	301,8721
|--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Bitmap1034],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,498980	220,3651

ANEXO 92

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 1 SELECT USANDO SUBQUERY + JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1020]=[open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]))	Compute Scalar	Compute Scalar	1905849	0	0,190585	538,29
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1905849	0	4,656106	538,0994
|--Hash Match(Right Outer Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open20].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]=[open20].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Right Outer Join	1905849	0	5,261154	533,4433
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,030326	0,03897994
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,000343	0,008653585
|--Compute Scalar(DEFINE:([Expr1016]=[open20].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	1905849	0	0,047646	528,1432
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1905849	0	5,509841	528,0955
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	1905849	0	4,330672	522,5857
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open20].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	1905849	0	3,504411	517,8057
|--Hash Match(Inner Join, HASH:([t].[id_ticket])=([l].[id_ticket]))	Hash Match	Inner Join	1905849	0	19,956590	514,3013
|--Bitmap(HASH:([t].[id_ticket]), DEFINE:([Opt_Bitmap1023]))	Bitmap	Bitmap Create	6666667	0	0,000000	83,86103
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	6666667	0	0,028502	83,86103
| |--Hash Match(Inner Join, HASH:([ps].[id_persona])=([t].[id_persona]))	Hash Match	Inner Join	6666667	0	32,425400	83,83253
| |--Bitmap(HASH:([ps].[id_persona]), DEFINE:([Bitmap1027]))	Bitmap	Bitmap Create	3	0	0,028506	0,03655184
| | |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	3	0	0,028506	0,03655184
| | |--Nested Loops(Inner Join, OUTER REFERENCES:([ps].[id_rol]))	Nested Loops	Inner Join	3	0	0,000042	0,0080456
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[persona].[persona_id_persona] AS [ps]))	Clustered Index Scan	Clustered Index Scan	10	0,003125	0,000168	0,003293
| | |--Clustered Index Seek(OBJECT:([open20].[dbo].[roles].[roles_id_rol] AS [r]), SEEK:([r].[id_rol]=[open20].[dbo].[persona].[id_rol] as [ps].[id_rol]), WHERE:([open20].[dbo].[roles].[nombre] as [r].[nombre]='Dependiente') ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,004706
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1027],[open20].[dbo].[tickets].[id_persona] as [t].[id_persona],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,500039	51,37057
|--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	1905849	0	7,770075	410,4836
|--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1022]))	Bitmap	Bitmap Create	1687233	0	1,039258	56,81057
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	1687233	0	1,039258	56,81057
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[ingresos].[ingresos_id_ingreso] AS [i]), WHERE:([open20].[dbo].[ingresos].[fecha] as [i].[fecha]>='2013-01-01 00:00:00.000' AND [open20].[dbo].[ingresos].[fecha] as [i].[fecha]<='2013-01-31 00:00:00.000'))	Clustered Index Scan	Clustered Index Scan	1687233	45,87127	5,500039	51,37131
|--Hash Match(Inner Join, HASH:([l].[id_ticket])=([im].[id_ingreso]))	Hash Match	Inner Join	1905849	0	33,190010	345,903
|--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	1687125	0	0,042178	235,9261
| |--Compute Scalar(DEFINE:([Expr1012]=CONVERT_IMPLICIT(numeric(10,0),[open20].[dbo].[lineasticket].[unidades] as [l].[unidades]*[open20].[dbo].[lineasticket].[precio] as [l].[precio],0)*(0.12)))	Compute Scalar	Compute Scalar	1687125	0	0,042178	235,9261
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	1687125	0	2,319721	235,8839
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1022],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1023],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,498980	220,3651
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([im].[id_ingreso]))	Parallelism	Repartition Streams	19971430	0	15,782220	76,78686
|--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [im]), WHERE:([open20].[dbo].[lineasimpuesto].[monto] as [im].[monto]<(500) AND PROBE([Bitmap1028],[open20].[dbo].[lineasimpuesto].[id_ingreso] as [im].[id_ingreso],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	19971430	53,10461	5,500039	58,60464

ANEXO 93

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 1 SELECT USANDO ÍNDICE + SUBQUERY 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1031]=[open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]))	Compute Scalar	Compute Scalar	4989357	0	0,498936	596,306
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	4989357	0	11,310060	595,807
|--Hash Match(Right Outer Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open20].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]=[open20].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Right Outer Join	4989357	0	13,816160	584,4969
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,030326	0,03897994
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,000343	0,008653585
|--Compute Scalar(DEFINE:([Expr1027]=[open20].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	4989357	0	0,124734	570,6418
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	4989357	0	13,431470	570,5171
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	4989357	0	10,219400	557,0856
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open20].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	4989357	0	8,181421	546,4169
|--Hash Match(Right Semi Join, HASH:([open20].[dbo].[tickets].[id_ticket])=([l].[id_ticket]))	Hash Match	Right Semi Join	4989357	0	23,144560	538,2355
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open20].[dbo].[tickets].[id_ticket]))	Parallelism	Repartition Streams	5549882	0	3,353226	57,55316
| |--Nested Loops(Inner Join, OUTER REFERENCES:([open20].[dbo].[persona].[id_persona]))	Nested Loops	Inner Join	5549882	0	5,799626	54,19994
| |--Sort(DISTINCT ORDER BY:([open20].[dbo].[persona].[id_persona] ASC))	Sort	Distinct Sort	3	0,002815315	0,000108	38,32022
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open20].[dbo].[persona].[id_persona]))	Parallelism	Repartition Streams	40	0	0,028510	38,3173
| | |--Hash Match(Partial Aggregate, HASH:([open20].[dbo].[persona].[id_persona]))	Hash Match	Partial Aggregate	40	0	0,653412	38,28879
| | |--Hash Match(Right Semi Join, HASH:([open20].[dbo].[ingresos].[id_ingreso])=([open20].[dbo].[tickets].[id_ticket]))	Hash Match	Right Semi Join	555692	0	12,131910	37,63538
| | |--Bitmap(HASH:([open20].[dbo].[ingresos].[id_ingreso]), DEFINE:([Bitmap1032]))	Bitmap	Bitmap Create	1667076	0	1,027183	3,779904
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open20].[dbo].[ingresos].[id_ingreso]))	Parallelism	Repartition Streams	1667076	0	1,027183	3,779904
| | | |--Index Seek(OBJECT:([open20].[dbo].[ingresos].[Ix_ingreso_fecha]), SEEK:([open20].[dbo].[ingresos].[fecha] >= '2013-01-01 00:00:00.000' AND [open20].[dbo].[ingresos].[fecha] <= '2013-01-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	1667076	2,294236	0,458485	2,752721
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open20].[dbo].[tickets].[id_ticket]), WHERE:(PROBE([Bitmap1032],[open20].[dbo].[tickets].[id_ticket])))	Parallelism	Repartition Streams	6666667	0	5,287250	21,72356
| | |--Nested Loops(Inner Join, OUTER REFERENCES:([open20].[dbo].[persona].[id_persona]) OPTIMIZED)	Nested Loops	Inner Join	6666667	0	5,573235	16,43631
| | |--Nested Loops(Inner Join, OUTER REFERENCES:([open20].[dbo].[persona].[id_rol]))	Nested Loops	Inner Join	3	0	0,000010	0,03654058
| | | |--Parallelism(Distribute Streams, RoundRobin Partitioning)	Parallelism	Distribute Streams	10	0	0,028526	0,03181932
| | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[persona].[persona_id_persona]))	Clustered Index Scan	Clustered Index Scan	10	0,003125	0,000168	0,003293
| | | |--Clustered Index Seek(OBJECT:([open20].[dbo].[roles].[roles_id_rol]), SEEK:([open20].[dbo].[roles].[id_rol]=[open20].[dbo].[persona].[id_rol]), WHERE:([open20].[dbo].[roles].[nombre]='Dependiente') ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,004706
| | |--Index Seek(OBJECT:([open20].[dbo].[tickets].[Ix_persona_tickets>]), SEEK:([open20].[dbo].[tickets].[id_persona]=[open20].[dbo].[persona].[id_persona]) ORDERED FORWARD)	Index Seek	Index Seek	1599972	1,613495	1,760126	10,82654
| |--Index Seek(OBJECT:([open20].[dbo].[tickets].[Ix_persona_tickets>]), SEEK:([open20].[dbo].[tickets].[id_persona]=[open20].[dbo].[persona].[id_persona]) ORDERED FORWARD)	Index Seek	Index Seek	2000000	2,015718	2,200157	10,08009
|--Hash Match(Right Semi Join, HASH:([open20].[dbo].[lineasimpuesto].[id_ingreso])=([l].[id_ticket]))	Hash Match	Right Semi Join	4989357	0	77,050510	457,5378
|--Bitmap(HASH:([open20].[dbo].[lineasimpuesto].[id_ingreso]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	1883201	0	0,000000	77,11522
| |--Hash Match(Inner Join, HASH:([open20].[dbo].[ingresos].[id_ingreso])=([open20].[dbo].[lineasimpuesto].[id_ingreso]))	Hash Match	Inner Join	1883201	0	28,376900	77,11522
| |--Bitmap(HASH:([open20].[dbo].[ingresos].[id_ingreso]), DEFINE:([Bitmap1033]))	Bitmap	Bitmap Create	1667076	0	1,027183	3,779904
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open20].[dbo].[ingresos].[id_ingreso]))	Parallelism	Repartition Streams	1667076	0	1,027183	3,779904
| | |--Index Seek(OBJECT:([open20].[dbo].[ingresos].[Ix_ingreso_fecha]), SEEK:([open20].[dbo].[ingresos].[fecha] >= '2013-01-01 00:00:00.000' AND [open20].[dbo].[ingresos].[fecha] <= '2013-01-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	1667076	2,294236	0,458485	2,752721
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open20].[dbo].[lineasimpuesto].[id_ingreso]))	Parallelism	Repartition Streams	19980870	0	11,998290	44,95841
| |--Index Seek(OBJECT:([open20].[dbo].[lineasimpuesto].[limpuesto_monto]), SEEK:([open20].[dbo].[lineasimpuesto].[monto] < (500)), WHERE:(PROBE([Bitmap1033],[open20].[dbo].[lineasimpuesto].[id_ingreso],N'[IN ROW]')) ORDERED FORWARD)	Index Seek	Index Seek	19980870	27,46535	5,494778	32,96012
|--Compute Scalar(DEFINE:([Expr1023]=CONVERT_IMPLICIT(numeric(10,0),[open20].[dbo].[lineasticket].[unidades] as [l].[unidades]*[open20].[dbo].[lineasticket].[precio] as [l].[precio],0)*(0.12)))	Compute Scalar	Compute Scalar	59996160	0	1,499904	303,372
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	59996160	0	81,507030	301,8721
|--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Bitmap1034],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,498980	220,3651

ANEXO 94

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 1 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1020]=[open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]))	Compute Scalar	Compute Scalar	1883081	0	0,188308	395,8755
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1883081	0	4,616477	395,6872
|--Hash Match(Right Outer Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open20].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]=[open20].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Right Outer Join	1883081	0	5,183076	391,0708
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,030326	0,03897994
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,000343	0,008653585
|--Compute Scalar(DEFINE:([Expr1016]=[open20].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	1883081	0	0,047077	385,8487
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1883081	0	5,444358	385,8016
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	1883081	0	4,287190	380,3573
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open20].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	1883081	0	3,462886	375,6208
|--Hash Match(Inner Join, HASH:([t].[id_ticket])=([l].[id_ticket]))	Hash Match	Inner Join	1883081	0	19,912530	372,1579
|--Bitmap(HASH:([t].[id_ticket]), DEFINE:([Opt_Bitmap1023]))	Bitmap	Bitmap Create	6666667	0	4,022250	23,06254
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	6666667	0	4,022250	23,06254
| |--Nested Loops(Inner Join, OUTER REFERENCES:([ps].[id_persona]) OPTIMIZED)	Nested Loops	Inner Join	6666667	0	6,966667	19,04029
| |--Nested Loops(Left Semi Join, WHERE:([open20].[dbo].[persona].[id_rol] as [ps].[id_rol]=[open20].[dbo].[roles].[id_rol] as [r].[id_rol]))	Nested Loops	Left Semi Join	3	0	0,000010	0,03653878
| | |--Parallelism(Distribute Streams, RoundRobin Partitioning)	Parallelism	Distribute Streams	10	0	0,028526	0,03181932
| | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[persona].[persona_id_persona] AS [ps]))	Clustered Index Scan	Clustered Index Scan	10	0,003125	0,000168	0,003293
| | |--Index Seek(OBJECT:([open20].[dbo].[roles].[roles_nombre] AS [r]), SEEK:([r].[nombre]='Dependiente') ORDERED FORWARD)	Index Seek	Index Seek	1	0,003125	0,000158	0,004706
| |--Index Seek(OBJECT:([open20].[dbo].[tickets].[Ix_persona_tickets>] AS [t]), SEEK:([t].[id_persona]=[open20].[dbo].[persona].[id_persona] as [ps].[id_persona]) ORDERED FORWARD)	Index Seek	Index Seek	2000000	2,015718	2,200157	12,03708
|--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	1883081	0	7,677461	329,1828
|--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1022]))	Bitmap	Bitmap Create	1667076	0	1,027183	3,779904
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	1667076	0	1,027183	3,779904
| |--Index Seek(OBJECT:([open20].[dbo].[ingresos].[Ix_ingreso_fecha] AS [i]), SEEK:([i].[fecha] >= '2013-01-01 00:00:00.000' AND [i].[fecha] <= '2013-01-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	1667076	2,294236	0,458485	2,752721
|--Hash Match(Inner Join, HASH:([l].[id_ticket])=([im].[id_ingreso]))	Hash Match	Inner Join	1883081	0	33,077400	317,7254
|--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1027]))	Bitmap	Bitmap Create	1666969	0	0,041674	235,8983
| |--Compute Scalar(DEFINE:([Expr1012]=CONVERT_IMPLICIT(numeric(10,0),[open20].[dbo].[lineasticket].[unidades] as [l].[unidades]*[open20].[dbo].[lineasticket].[precio] as [l].[precio],0)*(0.12)))	Compute Scalar	Compute Scalar	1666969	0	0,041674	235,8983
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	1666969	0	2,292348	235,8566
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1022],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1023],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,498980	220,3651
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([im].[id_ingreso]))	Parallelism	Repartition Streams	19980870	0	15,789660	48,74978
|--Index Seek(OBJECT:([open20].[dbo].[lineasimpuesto].[limpuesto_monto] AS [im]), SEEK:([im].[monto] < (500)), WHERE:(PROBE([Bitmap1027],[open20].[dbo].[lineasimpuesto].[id_ingreso] as [im].[id_ingreso],N'[IN ROW]')) ORDERED FORWARD)	Index Seek	Index Seek	19980870	27,46535	5,494778	32,96012

ANEXO 95

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 2 SELECT SIMPLE Y SELECT USANDO JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	140119	0	0,9304297	370,4639
|--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	140119	0	0,6910197	369,5335
|--Bitmap(HASH:([p].[id_producto]), DEFINE:([Opt_Bitmap1016]))	Bitmap	Bitmap Create	39382	0	0,1213837	0,8698564
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	39382	0	0,1213837	0,8698564
| |--Nested Loops(Inner Join, WHERE:([open20].[dbo].[categorias].[id_categoria] as [c].[id_categoria]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Nested Loops	Inner Join	39382	0	0,08230734	0,7484726
| |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	1	0	0,02850332	0,03178642
| | |--Index Seek(OBJECT:([open20].[dbo].[categorias].[categorias_nombre] AS [c]), SEEK:([c].[nombre]='FARMACEUTICOS') ORDERED FORWARD)	Index Seek	Index Seek	1	0,003125	0,0001581	0,0032831
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,02169907	0,61075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	135598	0	0,6828018	367,9726
|--Hash Match(Inner Join, HASH:([aci].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open20].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [aci].[id_atributoconjuntoinstancia]=[open20].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Inner Join	135598	0	0,3882457	367,2898
|--Bitmap(HASH:([aci].[id_atributoconjuntoinstancia]), DEFINE:([Opt_Bitmap1015]))	Bitmap	Bitmap Create	5	0	0,0285242	0,04017667
| |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	5	0	0,0285242	0,04017667
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [aci]), WHERE:([open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO NATURPHARMA' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO VITAFARMA'))	Clustered Index Scan	Clustered Index Scan	5	0,008310185	0,0013736	0,009683785
|--Hash Match(Inner Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]=[open20].[dbo].[tickets].[id_cliente] as [t].[id_cliente]))	Hash Match	Inner Join	135598	0	4,663817	366,8614
|--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1019]))	Bitmap	Bitmap Create	135598	0	0,2126503	324,6574
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	135598	0	0,2126503	324,6574
| |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	135598	0	23,51833	324,4448
| |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1018]))	Bitmap	Bitmap Create	135598	0	0,1869206	233,7511
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	135598	0	0,1869206	233,7511
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1015],[open20].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia],N'[IN ROW]') AND PROBE([Opt_Bitmap1016],[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto],N'[IN ROW]'))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,49898	220,3651
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	15,80475	67,17532
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1018],[open20].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]'))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,500039	51,37057
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	7,555548	37,54012
|--Clustered Index Scan(OBJECT:([open20].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1019],[open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,36831	0,6162642	29,98458

ANEXO 96

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 2 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	140119	0	0,9304297	142,505
|--Hash Match(Inner Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]=[open20].[dbo].[tickets].[id_cliente] as [t].[id_cliente]))	Hash Match	Inner Join	140119	0	7,401365	141,5746
|--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1019]))	Bitmap	Bitmap Create	140119	0	0,5668467	96,63306
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	140119	0	0,5668467	96,63306
| |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	140119	0	25,43117	96,06621
| |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1018]))	Bitmap	Bitmap Create	135646	0	0,5239223	3,476906
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	135646	0	0,5239223	3,476906
| | |--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	135646	0	0,6873977	2,952983
| | |--Bitmap(HASH:([p].[id_producto]), DEFINE:([Opt_Bitmap1014]))	Bitmap	Bitmap Create	39382	0	0,1213837	0,8698564
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	39382	0	0,1213837	0,8698564
| | | |--Nested Loops(Inner Join, WHERE:([open20].[dbo].[categorias].[id_categoria] as [c].[id_categoria]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Nested Loops	Inner Join	39382	0	0,08230734	0,7484726
| | | |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	1	0	0,02850332	0,03178642
| | | | |--Index Seek(OBJECT:([open20].[dbo].[categorias].[categorias_nombre] AS [c]), SEEK:([c].[nombre]='FARMACEUTICOS') ORDERED FORWARD)	Index Seek	Index Seek	1	0,003125	0,0001581	0,0032831
| | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,02169907	0,61075
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	135598	0	0,2875003	1,395726
| | |--Nested Loops(Inner Join, OUTER REFERENCES:([aci].[id_atributoconjuntoinstancia]))	Nested Loops	Inner Join	135598	0	0,1416997	1,108226
| | |--Parallelism(Repartition Streams, RoundRobin Partitioning)	Parallelism	Repartition Streams	5	0	0,02850605	0,03765181
| | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [aci]), WHERE:([open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO NATURPHARMA' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO VITAFARMA'))	Clustered Index Scan	Clustered Index Scan	5	0,008310185	0,0003434	0,008653585
| | |--Index Seek(OBJECT:([open20].[dbo].[lineasticket].[atribconjinst_descrip] AS [l]), SEEK:([l].[id_atributoconjuntoinstancia]=[open20].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [aci].[id_atributoconjuntoinstancia]), WHERE:(PROBE([Opt_Bitmap1014],[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto],N'[IN ROW]')) ORDERED FORWARD)	Index Seek	Index Seek	54246	0,1320139	0,05982769	0,8271899
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	15,80475	67,17532
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1018],[open20].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,500039	51,37057
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	7,555548	37,54012
|--Clustered Index Scan(OBJECT:([open20].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1019],[open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,36831	0,6162642	29,98458

ANEXO 97

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 2 SELECT USANDO SUBQUERY 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1017]=[Expr1031], [Expr1029]=[open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]))	Compute Scalar	Compute Scalar	135615	0	0,01356152	552,8507
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	135615	0	0,849582	552,8371
|--Hash Match(Right Outer Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]))	Hash Match	Right Outer Join	135615	0	0,286541	551,9875
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,02998294	0,03863652
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,0003434	0,008653585
|--Compute Scalar(DEFINE:([Expr1025]=[open20].[dbo].[productos].[precioventa] as [p].[precioventa]))	Compute Scalar	Compute Scalar	135615	0	0,00339038	551,6624
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	135615	0	0,8162293	551,659
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	135615	0	0,5057445	550,8428
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,09062924	0,7013792
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,02169907	0,61075
|--Compute Scalar(DEFINE:([Expr1021]=[open20].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	135615	0	0,00339038	549,6356
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	135615	0	0,9499679	549,6322
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,1585623	0,4493123
| |--Index Scan(OBJECT:([open20].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,02169907	0,29075
|--Filter(WHERE:([Expr1031]='FARMACEUTICOS'))	Filter	Filter	135615	0	0,03254765	548,233
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	271230	0	1,276448	548,2004
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	271230	0	0,2834358	546,924
|--Compute Scalar(DEFINE:([Expr1011]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	271230	0	0,00678076	456,8371
| |--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	271230	0	0,7647356	456,8303
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,09062924	0,7013792
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,02169907	0,61075
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	271230	0	1,145444	455,3642
| |--Hash Match(Inner Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open20].[dbo].[tickets].[id_cliente] as [t].[id_cliente]=[open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]))	Hash Match	Inner Join	271230	0	5,6488	454,2187
| |--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1036]))	Bitmap	Bitmap Create	271230	0	0	411,0298
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	271230	0	0,3968478	411,0298
| | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	271230	0	24,14407	410,6329
| | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1035]))	Bitmap	Bitmap Create	271230	0	0,3453819	319,3135
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	271230	0	0,3453819	319,3135
| | | |--Hash Match(Inner Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open20].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]=[open20].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]))	Hash Match	Inner Join	271230	0	98,56292	318,9682
| | | |--Bitmap(HASH:([a].[id_atributoconjuntoinstancia]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	5	0	0,02850937	0,04016183
| | | | |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	5	0	0,02850937	0,04016183
| | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]), WHERE:([open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO NATURPHARMA' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO VITAFARMA'))	Clustered Index Scan	Clustered Index Scan	5	0,008310185	0,0013736	0,009683785
| | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Bitmap1034],[open20].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,49898	220,3651
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	15,80475	67,17532
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1035],[open20].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,500039	51,37057
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	7,555548	37,54012
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1036],[open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,36831	0,6162642	29,98458
|--Assert(WHERE:(CASE WHEN [Expr1030]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	4,80E-07	89,80346
|--Stream Aggregate(DEFINE:([Expr1030]=Count(*), [Expr1031]=ANY([open20].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	1,10E-06	89,67327
|--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	4,18E-06	89,37492
|--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	45,35653
|--Clustered Index Seek(OBJECT:([open20].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	42,88465

ANEXO 98

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 2 SELECT USANDO SUBQUERY + JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1017]=[Expr1027], [Expr1025]=[open20].[dbo].[productos].[precioventa] as [p].[precioventa]))	Compute Scalar	Compute Scalar	135615	0	0,013562	461,5298
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	135615	0	0,894420	461,5162
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	135615	0	0,505745	460,6218
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,090629	0,7013792
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
|--Compute Scalar(DEFINE:([Expr1021]=[open20].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	135615	0	0,003390	459,4147
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	135615	0	0,949968	459,4113
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open20].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Filter(WHERE:([Expr1027]='FARMACEUTICOS'))	Filter	Filter	135615	0	0,032548	458,012
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	271230	0	1,468276	457,9795
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	271230	0	0,283436	456,5112
|--Compute Scalar(DEFINE:([Expr1011]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	271230	0	0,006781	366,4243
| |--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	271230	0	0,764736	366,4175
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,090629	0,7013792
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	271230	0	1,337271	364,9514
| |--Hash Match(Inner Join, HASH:([aci].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]), RESIDUAL:([open20].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [aci].[id_atributoconjuntoinstancia]=[open20].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia]))	Hash Match	Inner Join	271230	0	0,758775	363,6141
| |--Bitmap(HASH:([aci].[id_atributoconjuntoinstancia]), DEFINE:([Opt_Bitmap1028]))	Bitmap	Bitmap Create	5	0	0,028524	0,04017667
| | |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	5	0	0,028524	0,04017667
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [aci]), WHERE:([open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO NATURPHARMA' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO VITAFARMA'))	Clustered Index Scan	Clustered Index Scan	5	0,008310185	0,001374	0,009683785
| |--Hash Match(Inner Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]=[open20].[dbo].[tickets].[id_cliente] as [t].[id_cliente]))	Hash Match	Inner Join	271230	0	5,648800	362,8152
| |--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	271230	0	0,396848	319,6263
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	271230	0	0,396848	319,6263
| | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	271230	0	24,144070	319,2294
| | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1033]))	Bitmap	Bitmap Create	271230	0	0,345382	227,91
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	271230	0	0,345382	227,91
| | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1028],[open20].[dbo].[lineasticket].[id_atributoconjuntoinstancia] as [l].[id_atributoconjuntoinstancia],N'[IN ROW]'))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,498980	220,3651
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	15,804750	67,17532
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1033],[open20].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,500039	51,37057
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	7,555548	37,54012
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1034],[open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,36831	0,616264	29,98458
|--Assert(WHERE:(CASE WHEN [Expr1026]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	89,80346
|--Stream Aggregate(DEFINE:([Expr1026]=Count(*), [Expr1027]=ANY([open20].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	0,000001	89,67327
|--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,000004	89,37492
|--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	45,35653
|--Clustered Index Seek(OBJECT:([open20].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	42,88465

ANEXO 99

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 2 SELECT USANDO ÍNDICE + SUBQUERY 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1017]=[Expr1031], [Expr1029]=[open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]))	Compute Scalar	Compute Scalar	135615	0	0,013562	235,007
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	135615	0	0,854579	234,9934
|--Hash Match(Right Outer Join, HASH:([a].[id_atributoconjuntoinstancia])=([l].[id_atributoconjuntoinstancia]))	Hash Match	Right Outer Join	135615	0	0,286208	234,1388
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([a].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	1106	0	0,029983	0,03863652
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]))	Clustered Index Scan	Clustered Index Scan	1106	0,008310185	0,000343	0,008653585
|--Compute Scalar(DEFINE:([Expr1025]=[open20].[dbo].[productos].[precioventa] as [p].[precioventa]))	Compute Scalar	Compute Scalar	135615	0	0,003390	233,814
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_atributoconjuntoinstancia]))	Parallelism	Repartition Streams	135615	0	0,816229	233,8106
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	135615	0	0,505745	232,9944
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,090629	0,7013792
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
|--Compute Scalar(DEFINE:([Expr1021]=[open20].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	135615	0	0,003390	231,7872
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	135615	0	0,949968	231,7838
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open20].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Filter(WHERE:([Expr1031]='FARMACEUTICOS'))	Filter	Filter	135615	0	0,032548	230,3846
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	271230	0	1,276448	230,352
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	271230	0	0,283436	229,0756
|--Compute Scalar(DEFINE:([Expr1011]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	271230	0	0,006781	138,9887
| |--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	271230	0	0,764736	138,9819
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,090629	0,7013792
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	271230	0	1,145444	137,5158
| |--Hash Match(Inner Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open20].[dbo].[tickets].[id_cliente] as [t].[id_cliente]=[open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]))	Hash Match	Inner Join	271230	0	5,648800	136,3703
| |--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1035]))	Bitmap	Bitmap Create	271230	0	0,000000	93,1814
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	271230	0	0,396848	93,1814
| | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	271230	0	24,144070	92,78455
| | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	271230	0	0,345382	1,465153
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	271230	0	0,345382	1,465153
| | | |--Nested Loops(Inner Join, OUTER REFERENCES:([a].[id_atributoconjuntoinstancia]))	Nested Loops	Inner Join	271230	0	0,283436	1,119771
| | | |--Parallelism(Repartition Streams, RoundRobin Partitioning)	Parallelism	Repartition Streams	5	0	0,028502	0,0376481
| | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [a]), WHERE:([open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO NATURPHARMA' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [a].[descripcion]='ATRIB.CONJUNTO VITAFARMA'))	Clustered Index Scan	Clustered Index Scan	5	0,008310185	0,000343	0,008653585
| | | | |--Index Seek(OBJECT:([open20].[dbo].[lineasticket].[atribconjinst_descrip] AS [l]), SEEK:([l].[id_atributoconjuntoinstancia]=[open20].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [a].[id_atributoconjuntoinstancia]) ORDERED FORWARD)	Index Seek	Index Seek	54246	0,1320139	0,059828	0,8271899
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	15,804750	67,17532
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1034],[open20].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,500039	51,37057
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	7,555548	37,54012
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1035],[open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW')))	Clustered Index Scan	Clustered Index Scan	2240818	29,36831	0,616264	29,98458
|--Assert(WHERE:(CASE WHEN [Expr1030]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	89,80346
|--Stream Aggregate(DEFINE:([Expr1030]=Count(*), [Expr1031]=ANY([open20].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	0,000001	89,67327
|--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,000004	89,37492
|--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	45,35653
|--Clustered Index Seek(OBJECT:([open20].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	42,88465

ANEXO 100

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 2 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1017]=[Expr1027], [Expr1025]=[open20].[dbo].[productos].[precioventa] as [p].[precioventa]))	Compute Scalar	Compute Scalar	135615	0	0,013562	238,713
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	135615	0	0,894420	238,6995
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	135615	0	0,505745	237,8051
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,090629	0,7013792
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
|--Compute Scalar(DEFINE:([Expr1021]=[open20].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	135615	0	0,003390	236,5979
|--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	135615	0	0,949968	236,5945
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| |--Index Scan(OBJECT:([open20].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
|--Filter(WHERE:([Expr1027]='FARMACEUTICOS'))	Filter	Filter	135615	0	0,032548	235,1953
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	271230	0	1,468276	235,1627
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	271230	0	0,283436	233,6944
|--Compute Scalar(DEFINE:([Expr1011]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	271230	0	0,006781	143,6075
| |--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	271230	0	0,764736	143,6008
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,090629	0,7013792
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,021699	0,61075
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	271230	0	1,337271	142,1346
| |--Hash Match(Inner Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]=[open20].[dbo].[tickets].[id_cliente] as [t].[id_cliente]))	Hash Match	Inner Join	271230	0	8,016254	140,7974
| |--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	271230	0	0,598033	95,24098
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	271230	0	0,598033	95,24098
| | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	271230	0	25,801290	94,64295
| | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1033]))	Bitmap	Bitmap Create	271230	0	0,546567	1,666338
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	271230	0	0,546567	1,666338
| | | |--Nested Loops(Inner Join, OUTER REFERENCES:([aci].[id_atributoconjuntoinstancia]))	Nested Loops	Inner Join	271230	0	0,283436	1,119771
| | | |--Parallelism(Repartition Streams, RoundRobin Partitioning)	Parallelism	Repartition Streams	5	0	0,028506	0,03765181
| | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[atributoconjuntoinstancia].[atributoconjuntoinstancia_id_atributoconjuntoinstancia] AS [aci]), WHERE:([open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO NATURPHARMA' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR [open20].[dbo].[atributoconjuntoinstancia].[descripcion] as [aci].[descripcion]='ATRIB.CONJUNTO VITAFARMA'))	Clustered Index Scan	Clustered Index Scan	5	0,008310185	0,000343	0,008653585
| | | |--Index Seek(OBJECT:([open20].[dbo].[lineasticket].[atribconjinst_descrip] AS [l]), SEEK:([l].[id_atributoconjuntoinstancia]=[open20].[dbo].[atributoconjuntoinstancia].[id_atributoconjuntoinstancia] as [aci].[id_atributoconjuntoinstancia]) ORDERED FORWARD)	Index Seek	Index Seek	54246	0,1320139	0,059828	0,8271899
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	15,804750	67,17532
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1033],[open20].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,500039	51,37057
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	7,555548	37,54012
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1034],[open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,36831	0,616264	29,98458
|--Assert(WHERE:(CASE WHEN [Expr1026]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	89,80346
|--Stream Aggregate(DEFINE:([Expr1026]=Count(*), [Expr1027]=ANY([open20].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	0,000001	89,67327
|--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,000004	89,37492
|--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	45,35653
|--Clustered Index Seek(OBJECT:([open20].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	42,88465

ANEXO 101

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 3 SELECT SIMPLE Y SELECT USANDO JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,02850967	1568,235
|--Filter(WHERE:([Expr1010]>=(600) AND [Expr1010]<=(1000)))	Filter	Filter	1	0	3,886566	1568,206
|--Stream Aggregate(GROUP BY:([l].[id_producto], [l].[precio], [c].[nombre]) DEFINE:([Expr1010]=SUM([open20].[dbo].[lineasticket].[unidades] as [l].[unidades]), [p].[nombre]=ANY([open20].[dbo].[productos].[nombre] as [p].[nombre]), [c].[id_categoria]=ANY([open20].[dbo].[categorias].[id_categoria] as [c].[id_categoria])))	Stream Aggregate	Aggregate	17666210	0	5,214305	1564,32
|--Sort(ORDER BY:([l].[id_producto] ASC, [l].[precio] ASC, [c].[nombre] ASC))	Sort	Sort	20040190	805,9938	266,2565	1559,105
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto], [l].[precio], [c].[nombre]))	Parallelism	Repartition Streams	20040190	0	54,97119	486,8551
|--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	20040190	0	39,07379	431,8839
|--Bitmap(HASH:([p].[id_producto]), DEFINE:([Opt_Bitmap1012]))	Bitmap	Bitmap Create	78763	0	0,3701747	1,267548
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,3701747	1,267548
| |--Hash Match(Inner Join, HASH:([c].[id_categoria])=([p].[id_categoria]))	Hash Match	Inner Join	78763	0	0,2548291	0,897373
| |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	2	0	0,02850664	0,03179085
| | |--Index Scan(OBJECT:([open20].[dbo].[categorias].[categorias_nombre] AS [c]))	Index Scan	Index Scan	2	0,003125	0,0001592	0,0032842
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,02169907	0,61075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	20040190	0	19,63908	391,5425
|--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	20040190	0	55,09517	371,9035
|--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1011]))	Bitmap	Bitmap Create	6681348	0	4,031045	59,80236
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	6681348	0	4,031045	59,80236
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[ingresos].[ingresos_id_ingreso] AS [i]), WHERE:([open20].[dbo].[ingresos].[fecha] as [i].[fecha]>='2013-05-01 00:00:00.000' AND [open20].[dbo].[ingresos].[fecha] as [i].[fecha]<='2013-08-31 00:00:00.000'))	Clustered Index Scan	Clustered Index Scan	6681348	45,87127	5,500039	51,37131
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	20040190	0	23,44171	257,0059
|--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1011],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1012],[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,49898	220,3651

ANEXO 102

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 3 SELECT ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,02850967	1520,511
|--Filter(WHERE:([Expr1010]>=(600) AND [Expr1010]<=(1000)))	Filter	Filter	1	0	3,879007	1520,482
|--Stream Aggregate(GROUP BY:([l].[id_producto], [l].[precio], [c].[nombre]) DEFINE:([Expr1010]=SUM([open20].[dbo].[lineasticket].[unidades] as [l].[unidades]), [p].[nombre]=ANY([open20].[dbo].[productos].[nombre] as [p].[nombre]), [c].[id_categoria]=ANY([open20].[dbo].[categorias].[id_categoria] as [c].[id_categoria]))	Stream Aggregate	Aggregate	17631850	0	5,203242	1516,603
|--Sort(ORDER BY:([l].[id_producto] ASC, [l].[precio] ASC, [c].[nombre] ASC))	Sort	Sort	19995070	804,1779	265,5824	1511,4
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto], [l].[precio], [c].[nombre]))	Parallelism	Repartition Streams	19995070	0	54,84749	441,6395
|--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	19995070	0	38,98762	386,792
|--Bitmap(HASH:([p].[id_producto]), DEFINE:([Opt_Bitmap1012]))	Bitmap	Bitmap Create	78763	0	0,3701747	1,267548
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,3701747	1,267548
| |--Hash Match(Inner Join, HASH:([c].[id_categoria])=([p].[id_categoria]))	Hash Match	Inner Join	78763	0	0,2548291	0,897373
| |--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	2	0	0,02850664	0,03179085
| | |--Index Scan(OBJECT:([open20].[dbo].[categorias].[categorias_nombre] AS [c]))	Index Scan	Index Scan	2	0,003125	0,0001592	0,0032842
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]))	Clustered Index Scan	Clustered Index Scan	78763	0,5890509	0,02169907	0,61075
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	19995070	0	19,59493	346,5368
|--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	19995070	0	54,96803	326,9419
|--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1011]))	Bitmap	Bitmap Create	6666306	0	4,022034	15,02065
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	6666306	0	4,022034	15,02065
| |--Index Seek(OBJECT:([open20].[dbo].[ingresos].[Ix_ingresos_fecha] AS [i]), SEEK:([i].[fecha] >= '2013-05-01 00:00:00.000' AND [i].[fecha] <= '2013-08-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	6666306	9,165347	1,833273	10,99862
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	19995070	0	23,38899	256,9532
|--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1011],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]') AND PROBE([Opt_Bitmap1012],[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,49898	220,3651

ANEXO 103

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 3 SELECT SUBQUERY 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtreeCost
|--Compute Scalar(DEFINE:([Expr1018]=[Expr1024], [Expr1022]=[open20].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	1	0	0,0000001	774,4235
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,0285095	774,4235
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,0000010	774,395
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,0000010	774,3917
| |--Compute Scalar(DEFINE:([Expr1012]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	1	0	0,0000000	774,3851
| | |--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,0000010	774,3851
| | |--Filter(WHERE:([Expr1008]>=(600) AND [Expr1008]<=(1000)))	Filter	Filter	1	0	3,1010170	774,3818
| | | |--Hash Match(Aggregate, HASH:([l].[id_producto], [l].[precio]), RESIDUAL:([open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto] = [open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto] AND [open20].[dbo].[lineasticket].[precio] as [l].[precio] = [open20].[dbo].[lineasticket].[precio] as [l].[precio]) DEFINE:([Expr1008]=SUM([open20].[dbo].[lineasticket].[unidades] as [l].[unidades]))	Hash Match	Aggregate	14095530	81,46744	134,3367000	771,2808
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto], [l].[precio]))	Parallelism	Repartition Streams	20042760	0	20,8967700	555,4766
| | | |--Hash Match(Inner Join, HASH:([open20].[dbo].[ingresos].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	20042760	0	101,8333000	534,5798
| | | |--Bitmap(HASH:([open20].[dbo].[ingresos].[id_ingreso]), DEFINE:([Bitmap1029]))	Bitmap	Bitmap Create	6681348	0	44,8152900	142,2587
| | | | |--Hash Match(Inner Join, HASH:([open20].[dbo].[ingresos].[id_ingreso])=([open20].[dbo].[tickets].[id_ticket]))	Hash Match	Inner Join	6681348	0	44,8152900	142,2587
| | | | |--Bitmap(HASH:([open20].[dbo].[ingresos].[id_ingreso]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	6681348	0	4,0310450	59,80236
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open20].[dbo].[ingresos].[id_ingreso]))	Parallelism	Repartition Streams	6681348	0	4,0310450	59,80236
| | | | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[ingresos].[ingresos_id_ingreso]), WHERE:([open20].[dbo].[ingresos].[fecha]>='2013-05-01 00:00:00.000' AND [open20].[dbo].[ingresos].[fecha]<='2013-08-31 00:00:00.000'))	Clustered Index Scan	Clustered Index Scan	6681348	45,87127	5,5000390	51,37131
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open20].[dbo].[tickets].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	12,0097500	37,64106
| | | | |--Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_ticketipo]), WHERE:(PROBE([Bitmap1028],[open20].[dbo].[tickets].[id_ticket],N'[IN ROW]')))	Index Scan	Index Scan	20000000	20,13127	5,5000390	25,63131
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	59996160	0	70,1227600	290,4878
| | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Bitmap1029],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,4989800	220,3651
| |--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831
| |--Assert(WHERE:(CASE WHEN [Expr1023]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,0000005	0,00657196
| |--Stream Aggregate(DEFINE:([Expr1023]=Count(*), [Expr1024]=ANY([open20].[dbo].[categorias].[nombre] as [c].[nombre]))	Stream Aggregate	Aggregate	1	0	0,0000011	0,00657148
| |--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,0000042	0,00657038
| |--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831
| |--Clustered Index Seek(OBJECT:([open20].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831
|--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831

ANEXO 104

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 3 SELECT SUBQUERY + JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1018]=[Expr1020]))	Compute Scalar	Compute Scalar	1	0	0,0000001	1081,868
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,0285077	1081,868
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,0000010	1081,839
|--Compute Scalar(DEFINE:([Expr1012]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	1	0	0,0000000	1081,833
| |--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,0000010	1081,833
| |--Filter(WHERE:([Expr1008]>=(600) AND [Expr1008]<=(1000)))	Filter	Filter	1	0	3,1004890	1081,829
| | |--Hash Match(Aggregate, HASH:([p].[nombre], [l].[precio]), RESIDUAL:([open20].[dbo].[productos].[nombre] as [p].[nombre] = [open20].[dbo].[productos].[nombre] as [p].[nombre] AND [open20].[dbo].[lineasticket].[precio] as [l].[precio] = [open20].[dbo].[lineasticket].[precio] as [l].[precio]) DEFINE:([Expr1008]=SUM([open20].[dbo].[lineasticket].[unidades] as [l].[unidades]), [l].[id_producto]=ANY([open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]))	Hash Match	Aggregate	14093130	342,8075	265,7442000	1078,729
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[nombre], [l].[precio]))	Parallelism	Repartition Streams	20040190	0	45,2429300	470,1774
| | |--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	20040190	0	38,9656700	424,9344
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,1245958	0,4153458
| | | |--Index Scan(OBJECT:([open20].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,0216991	0,29075
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	20042760	0	19,6415900	385,5534
| | |--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	20042760	0	55,1001500	365,9118
| | |--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1021]))	Bitmap	Bitmap Create	6681348	0	4,0310450	59,80236
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	6681348	0	4,0310450	59,80236
| | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[ingresos].[ingresos_id_ingreso] AS [i]), WHERE:([open20].[dbo].[ingresos].[fecha] as [i].[fecha]>='2013-05-01 00:00:00.000' AND [open20].[dbo].[ingresos].[fecha] as [i].[fecha]<='2013-08-31 00:00:00.000'))	Clustered Index Scan	Clustered Index Scan	6681348	45,87127	5,5000390	51,37131
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	20042760	0	23,4447100	251,0093
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1021],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,4989800	220,3651
| |--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831
|--Assert(WHERE:(CASE WHEN [Expr1019]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,0000005	0,00657196
|--Stream Aggregate(DEFINE:([Expr1019]=Count(*), [Expr1020]=ANY([open20].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	0,0000011	0,00657148
|--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,0000042	0,00657038
|--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831
|--Clustered Index Seek(OBJECT:([open20].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831

ANEXO 105

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 3 SELECT ÍNDICE + SUBQUERY 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1022]=[Expr1024]))	Compute Scalar	Compute Scalar	1	0	0,0000001	728,9222
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,0285095	728,9222
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,0000010	728,8937
|--Compute Scalar(DEFINE:([Expr1016]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	1	0	0,0000000	728,8871
| |--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,0000010	728,8871
| |--Compute Scalar(DEFINE:([Expr1012]=[open20].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	1	0	0,0000000	728,8838
| | |--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,0000010	728,8838
| | |--Filter(WHERE:([Expr1008]>=(600) AND [Expr1008]<=(1000)))	Filter	Filter	1	0	3,0963360	728,8805
| | | |--Hash Match(Aggregate, HASH:([l].[id_producto], [l].[precio]), RESIDUAL:([open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto] = [open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto] AND [open20].[dbo].[lineasticket].[precio] as [l].[precio] = [open20].[dbo].[lineasticket].[precio] as [l].[precio]) DEFINE:([Expr1008]=SUM([open20].[dbo].[lineasticket].[unidades] as [l].[unidades])))	Hash Match	Aggregate	14074260	81,17863	134,0896000	725,7842
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto], [l].[precio]))	Parallelism	Repartition Streams	19997640	0	20,8497900	510,516
| | | |--Hash Match(Inner Join, HASH:([open20].[dbo].[ingresos].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	19997640	0	101,7527000	489,6662
| | | |--Bitmap(HASH:([open20].[dbo].[ingresos].[id_ingreso]), DEFINE:([Bitmap1029]))	Bitmap	Bitmap Create	6666306	0	44,7639600	97,42568
| | | | |--Hash Match(Inner Join, HASH:([open20].[dbo].[ingresos].[id_ingreso])=([open20].[dbo].[tickets].[id_ticket]))	Hash Match	Inner Join	6666306	0	44,7639600	97,42568
| | | | |--Bitmap(HASH:([open20].[dbo].[ingresos].[id_ingreso]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	6666306	0	4,0220340	15,02065
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open20].[dbo].[ingresos].[id_ingreso]))	Parallelism	Repartition Streams	6666306	0	4,0220340	15,02065
| | | | | | |--Index Seek(OBJECT:([open20].[dbo].[ingresos].[Ix_ingresos_fecha]), SEEK:([open20].[dbo].[ingresos].[fecha] >= '2013-05-01 00:00:00.000' AND [open20].[dbo].[ingresos].[fecha] <= '2013-08-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	6666306	9,165347	1,8332730	10,99862
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([open20].[dbo].[tickets].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	12,0097500	37,64106
| | | | |--Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_ticketipo]), WHERE:(PROBE([Bitmap1028],[open20].[dbo].[tickets].[id_ticket],N'[IN ROW]')))	Index Scan	Index Scan	20000000	20,13127	5,5000390	25,63131
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	59996160	0	70,1227600	290,4878
| | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Bitmap1029],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,4989800	220,3651
| | |--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831
| |--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831
|--Assert(WHERE:(CASE WHEN [Expr1023]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,0000005	0,00657196
|--Stream Aggregate(DEFINE:([Expr1023]=Count(*), [Expr1024]=ANY([open20].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	0,0000011	0,00657148
|--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,0000042	0,00657038
|--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831
|--Clustered Index Seek(OBJECT:([open20].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831

ANEXO 106

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 3 SELECT ÍNDICE + SUBQUERY + JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1018]=[Expr1020]))	Compute Scalar	Compute Scalar	1	0	0,0000001	1035,328
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,0285077	1035,328
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,0000010	1035,299
|--Compute Scalar(DEFINE:([Expr1012]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]))	Compute Scalar	Compute Scalar	1	0	0,0000000	1035,293
| |--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,0000010	1035,293
| |--Filter(WHERE:([Expr1008]>=(600) AND [Expr1008]<=(1000)))	Filter	Filter	1	0	3,0958090	1035,29
| | |--Hash Match(Aggregate, HASH:([p].[nombre], [l].[precio]), RESIDUAL:([open20].[dbo].[productos].[nombre] as [p].[nombre] = [open20].[dbo].[productos].[nombre] as [p].[nombre] AND [open20].[dbo].[lineasticket].[precio] as [l].[precio] = [open20].[dbo].[lineasticket].[precio] as [l].[precio]) DEFINE:([Expr1008]=SUM([open20].[dbo].[lineasticket].[unidades] as [l].[unidades]), [l].[id_producto]=ANY([open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto])))	Hash Match	Aggregate	14071860	341,9303	265,2798000	1032,194
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[nombre], [l].[precio]))	Parallelism	Repartition Streams	19995070	0	45,1411300	424,9837
| | |--Hash Match(Inner Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Inner Join	19995070	0	38,8795000	379,8425
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,1245958	0,4153458
| | | |--Index Scan(OBJECT:([open20].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,0216991	0,29075
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	19997640	0	19,5974400	340,5477
| | |--Hash Match(Inner Join, HASH:([i].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	19997640	0	54,9730000	320,9503
| | |--Bitmap(HASH:([i].[id_ingreso]), DEFINE:([Opt_Bitmap1021]))	Bitmap	Bitmap Create	6666306	0	4,0220340	15,02065
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	6666306	0	4,0220340	15,02065
| | | |--Index Seek(OBJECT:([open20].[dbo].[ingresos].[Ix_ingresos_fecha] AS [i]), SEEK:([i].[fecha] >= '2013-05-01 00:00:00.000' AND [i].[fecha] <= '2013-08-31 00:00:00.000') ORDERED FORWARD)	Index Seek	Index Seek	6666306	9,165347	1,8332730	10,99862
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	19997640	0	23,3919900	250,9566
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1021],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,4989800	220,3651
| |--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831
|--Assert(WHERE:(CASE WHEN [Expr1019]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,0000005	0,00657196
|--Stream Aggregate(DEFINE:([Expr1019]=Count(*), [Expr1020]=ANY([open20].[dbo].[categorias].[nombre] as [c].[nombre])))	Stream Aggregate	Aggregate	1	0	0,0000011	0,00657148
|--Nested Loops(Inner Join, OUTER REFERENCES:([p].[id_categoria]))	Nested Loops	Inner Join	1	0	0,0000042	0,00657038
|--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831
|--Clustered Index Seek(OBJECT:([open20].[dbo].[categorias].[categorias_id_categoria] AS [c]), SEEK:([c].[id_categoria]=[open20].[dbo].[productos].[id_categoria] as [p].[id_categoria]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831

ANEXO 107

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 4 SELECT SIMPLE Y SELECT USANDO JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,028517	626,0939
|--Nested Loops(Inner Join, OUTER REFERENCES:([t].[id_cliente]) OPTIMIZED)	Nested Loops	Inner Join	1	0	0,000001	626,0654
|--Nested Loops(Inner Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Inner Join	1	0	0,000001	626,0621
| |--Nested Loops(Inner Join, OUTER REFERENCES:([t].[id_persona]))	Nested Loops	Inner Join	1	0	0,000001	626,0588
| | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([t].[id_ticket]))	Hash Match	Inner Join	1	0	31,708780	626,0555
| | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	1539246	0	31,440950	523,3764
| | | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([i].[id_ingreso]))	Hash Match	Inner Join	1539246	0	31,440950	523,3764
| | | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1027]))	Bitmap	Bitmap Create	1539246	0	2,410965	423,3794
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	1539246	0	2,410965	423,3794
| | | | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso], [l].[id_impuesto])=([li].[id_ingreso], [li].[id_impuesto]), RESIDUAL:([open20].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso]=[open20].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso] AND [open20].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]=[open20].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto]))	Hash Match	Inner Join	1539246	0	44,409860	420,9684
| | | | | | |--Bitmap(HASH:([pg].[id_ingreso], [l].[id_impuesto]), DEFINE:([Bitmap1026]))	Bitmap	Bitmap Create	1352681	0	2,206908	297,1017
| | | | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso], [l].[id_impuesto]))	Parallelism	Repartition Streams	1352681	0	2,206908	297,1017
| | | | | | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	1352681	0	3,912089	294,8948
| | | | | | | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Opt_Bitmap1016]))	Bitmap	Bitmap Create	450922	0	0,384193	61,55254
| | | | | | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	450922	0	0,384193	61,55254
| | | | | | | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[pagos].[pagos_id_pagos] AS [pg]), WHERE:([open20].[dbo].[pagos].[total] as [pg].[total]>=(500) AND [open20].[dbo].[pagos].[total] as [pg].[total]<=(800)))	Clustered Index Scan	Clustered Index Scan	450922	51,26831	5,500039	56,76835
| | | | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	1352681	0	1,865525	229,4301
| | | | | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1016],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,498980	220,3651
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([li].[id_ingreso], [li].[id_impuesto]))	Parallelism	Repartition Streams	20000000	0	20,852250	79,45689
| | | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [li]), WHERE:(PROBE([Bitmap1026],[open20].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso],[open20].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto])))	Clustered Index Scan	Clustered Index Scan	20000000	53,10461	5,500039	58,60464
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	20000000	0	17,184750	68,55606
| | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[ingresos].[ingresos_id_ingreso] AS [i]), WHERE:(PROBE([Bitmap1027],[open20].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87127	5,500039	51,37131
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	19,599750	70,97032
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1028],[open20].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,500039	51,37057
| |--Clustered Index Seek(OBJECT:([open20].[dbo].[persona].[persona_id_persona] AS [per]), SEEK:([per].[id_persona]=[open20].[dbo].[tickets].[id_persona] as [t].[id_persona]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
|--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
|--Clustered Index Seek(OBJECT:([open20].[dbo].[clientes].[clientes_id_cliente] AS [cl]), SEEK:([cl].[id_cliente]=[open20].[dbo].[tickets].[id_cliente] as [t].[id_cliente]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831

ANEXO 108

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 4 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,028517	564,954
|--Nested Loops(Inner Join, OUTER REFERENCES:([t].[id_cliente]) OPTIMIZED)	Nested Loops	Inner Join	1	0	0,000001	564,9255
|--Nested Loops(Inner Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Inner Join	1	0	0,000001	564,9222
|--Nested Loops(Inner Join, OUTER REFERENCES:([t].[id_persona]))	Nested Loops	Inner Join	1	0	0,000001	564,919
|--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([t].[id_ticket]))	Hash Match	Inner Join	1	0	31,552420	564,9157
|--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	1511945	0	31,289340	462,3929
|--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([i].[id_ingreso]))	Hash Match	Inner Join	1511945	0	31,289340	462,3929
|--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1027]))	Bitmap	Bitmap Create	1511945	0	2,368708	362,5475
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	1511945	0	2,368708	362,5475
|--Hash Match(Inner Join, HASH:([pg].[id_ingreso], [l].[id_impuesto])=([li].[id_ingreso], [li].[id_impuesto]), RESIDUAL:([open20].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso]=[open20].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso] AND [open20].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]=[open20].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto]))	Hash Match	Inner Join	1511945	0	44,202100	360,1788
|--Bitmap(HASH:([pg].[id_ingreso], [l].[id_impuesto]), DEFINE:([Bitmap1026]))	Bitmap	Bitmap Create	1328688	0	2,168269	236,5198
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso], [l].[id_impuesto]))	Parallelism	Repartition Streams	1328688	0	2,168269	236,5198
|--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	1328688	0	3,843015	234,3516
|--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Opt_Bitmap1016]))	Bitmap	Bitmap Create	442924	0	0,377884	1,111001
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	442924	0	0,377884	1,111001
|--Index Seek(OBJECT:([open20].[dbo].[pagos].[pagos_total] AS [pg]), SEEK:([pg].[total] >= (500) AND [pg].[total] <= (800)) ORDERED FORWARD)	Index Seek	Index Seek	442924	0,6112732	0,121844	0,7331166
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	1328688	0	1,832942	229,3976
|--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1016],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,498980	220,3651
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([li].[id_ingreso], [li].[id_impuesto]))	Parallelism	Repartition Streams	20000000	0	20,852250	79,45689
|--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [li]), WHERE:(PROBE([Bitmap1026],[open20].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso],[open20].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto])))	Clustered Index Scan	Clustered Index Scan	20000000	53,10461	5,500039	58,60464
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	20000000	0	17,184750	68,55606
|--Clustered Index Scan(OBJECT:([open20].[dbo].[ingresos].[ingresos_id_ingreso] AS [i]), WHERE:(PROBE([Bitmap1027],[open20].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87127	5,500039	51,37131
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	19,599750	70,97032
|--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1028],[open20].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,500039	51,37057
|--Clustered Index Seek(OBJECT:([open20].[dbo].[persona].[persona_id_persona] AS [per]), SEEK:([per].[id_persona]=[open20].[dbo].[tickets].[id_persona] as [t].[id_persona]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
|--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
|--Clustered Index Seek(OBJECT:([open20].[dbo].[clientes].[clientes_id_cliente] AS [cl]), SEEK:([cl].[id_cliente]=[open20].[dbo].[tickets].[id_cliente] as [t].[id_cliente]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831

ANEXO 109

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 4 SELECT USANDO SUBQUERY 20 MILLONES DE DATOS..

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1019]=[Expr1025], [Expr1023]=[open20].[dbo].[persona].[nombre] as [per].[nombre]))	Compute Scalar	Compute Scalar	1352681	0	0,135268	2456,899
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1352681	0	0,028517	2456,763
|--Hash Match(Right Outer Join, HASH:([per].[id_persona])=([t].[id_persona]))	Hash Match	Right Outer Join	1352681	0	2,900435	2456,735
|--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	10	0	0,028544	0,03183658
| |--Index Scan(OBJECT:([open20].[dbo].[personal].[persona_nombre] AS [per]))	Index Scan	Index Scan	10	0,003125	0,000168	0,003293
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([i].[id_ingreso], [l].[id_impuesto]))	Nested Loops	Left Outer Join	1352681	0	1,413551	2453,803
|--Compute Scalar(DEFINE:([Expr1015]=[open20].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	1352681	0	0,033817	558,9247
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso], [l].[id_impuesto]))	Parallelism	Repartition Streams	1352681	0	5,636967	558,8909
| |--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	1352681	0	3,274258	553,254
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| | |--Index Scan(OBJECT:([open20].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
| |--Compute Scalar(DEFINE:([Expr1011]=[open20].[dbo].[clientes].[nombres] as [cl].[nombres]))	Compute Scalar	Compute Scalar	1352681	0	0,033817	549,5304
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	1352681	0	4,432236	549,4966
| |--Hash Match(Left Outer Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]=[open20].[dbo].[tickets].[id_cliente] as [t].[id_cliente]))	Hash Match	Left Outer Join	1352681	0	18,965610	545,0643
| |--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1036]))	Bitmap	Bitmap Create	1352681	0	3,265549	492,3853
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	1352681	0	3,265549	492,3853
| | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([t].[id_ticket]))	Hash Match	Inner Join	1352681	0	24,767230	489,1198
| | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1035]))	Bitmap	Bitmap Create	135268	0	0,162322	393,3823
| | | |--Filter(WHERE:([open20].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso]=[open20].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso]))	Filter	Filter	135268	0	0,162322	393,3823
| | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	1352681	0	3,912089	393,2199
| | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Opt_Bitmap1027]))	Bitmap	Bitmap Create	450922	0	0,384193	61,55254
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	450922	0	0,384193	61,55254
| | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[pagos].[pagos_id_pagos] AS [pg]), WHERE:([open20].[dbo].[pagos].[total] as [pg].[total]>=(500) AND [open20].[dbo].[pagos].[total] as [pg].[total]<=(800)))	Clustered Index Scan	Clustered Index Scan	450922	51,26831	5,500039	56,76835
| | | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([i].[id_ingreso]))	Hash Match	Inner Join	1352681	0	29,769100	327,7553
| | | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	1352681	0	1,865525	229,4301
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	1352681	0	1,865525	229,4301
| | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1027],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,498980	220,3651
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	20000000	0	17,184750	68,55606
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[ingresos].[ingresos_id_ingreso] AS [i]), WHERE:(PROBE([Bitmap1034],[open20].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87127	5,500039	51,37131
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	19,599750	70,97032
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1035],[open20].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,500039	51,37057
|--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	3,728791	33,71336
|--Clustered Index Scan(OBJECT:([open20].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1036],[open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,36831	0,616264	29,98458
|--Index Spool(SEEK:([i].[id_ingreso]=[open20].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso] AND [l].[id_impuesto]=[open20].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]))	Index Spool	Lazy Spool	1	0,003125	0,000258	1893,464
|--Assert(WHERE:(CASE WHEN [Expr1024]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	1539,155
|--Stream Aggregate(DEFINE:([Expr1024]=Count(*), [Expr1025]=ANY([open20].[dbo].[lineasimpuesto].[monto] as [li].[monto])))	Stream Aggregate	Aggregate	1	0	0,000001	1538,939
|--Index Spool(SEEK:([li].[id_ingreso]=[open20].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso] AND [li].[id_impuesto]=[open20].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]))	Index Spool	Eager Spool	1	453,4861	20,000260	1538,402
|--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [li]))	Clustered Index Scan	Clustered Index Scan	20000000	53,10461	22,000160	75,10477

ANEXO 110

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 4 SELECT USANDO SUBQUERY + JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1021]=[open20].[dbo].[persona].[nombre] as [per].[nombre]))	Compute Scalar	Compute Scalar	1	0	0,000000	590,4825
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,028517	590,4825
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([t].[id_persona]))	Nested Loops	Left Outer Join	1	0	0,000001	590,454
|--Compute Scalar(DEFINE:([Expr1017]=[open20].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	1	0	0,000000	590,4507
| |--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,000001	590,4507
| |--Compute Scalar(DEFINE:([Expr1013]=[open20].[dbo].[clientes].[nombres] as [cl].[nombres]))	Compute Scalar	Compute Scalar	1	0	0,000000	590,4474
| | |--Nested Loops(Left Outer Join, OUTER REFERENCES:([t].[id_cliente]) OPTIMIZED)	Nested Loops	Left Outer Join	1	0	0,000001	590,4474
| | |--Nested Loops(Inner Join, OUTER REFERENCES:([l].[id_ticket], [Expr1030]) WITH UNORDERED PREFETCH)	Nested Loops	Inner Join	1	0	0,160860	590,4442
| | | |--Sort(ORDER BY:([l].[id_ticket] ASC))	Sort	Sort	153933	0,002815315	3,043852	515,788
| | | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso], [l].[id_impuesto])=([li].[id_ingreso], [li].[id_impuesto]), RESIDUAL:([open20].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso]=[open20].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso] AND [open20].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]=[open20].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto]))	Hash Match	Inner Join	153933	0	34,230980	512,7413
| | | | |--Bitmap(HASH:([pg].[id_ingreso], [l].[id_impuesto]), DEFINE:([Bitmap1029]))	Bitmap	Bitmap Create	135268	0	0,162322	399,0534
| | | | | |--Filter(WHERE:([open20].[dbo].[tickets].[id_ticket] as [t].[id_ticket]=[open20].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso]))	Filter	Filter	135268	0	0,162322	399,0534
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso], [l].[id_impuesto]))	Parallelism	Repartition Streams	1352681	0	3,256926	398,8911
| | | | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	1352681	0	3,912089	395,6342
| | | | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Opt_Bitmap1026]))	Bitmap	Bitmap Create	450922	0	0,384193	61,55254
| | | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	450922	0	0,384193	61,55254
| | | | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[pagos].[pagos_id_pagos] AS [pg]), WHERE:([open20].[dbo].[pagos].[total] as [pg].[total]>=(500) AND [open20].[dbo].[pagos].[total] as [pg].[total]<=(800)))	Clustered Index Scan	Clustered Index Scan	450922	51,26831	5,500039	56,76835
| | | | | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	1352681	0	29,769100	330,1696
| | | | | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	1352681	0	1,865525	229,4301
| | | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	1352681	0	1,865525	229,4301
| | | | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1026],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,498980	220,3651
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	19,599750	70,97032
| | | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1028],[open20].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,500039	51,37057
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([li].[id_ingreso], [li].[id_impuesto]))	Parallelism	Repartition Streams	20000000	0	20,852250	79,45689
| | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [li]), WHERE:(PROBE([Bitmap1029],[open20].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso],[open20].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto])))	Clustered Index Scan	Clustered Index Scan	20000000	53,10461	5,500039	58,60464
| | | |--Clustered Index Seek(OBJECT:([open20].[dbo].[ingresos].[ingresos_id_ingreso] AS [i]), SEEK:([i].[id_ingreso]=[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	74,49534
| | |--Clustered Index Seek(OBJECT:([open20].[dbo].[clientes].[clientes_id_cliente] AS [cl]), SEEK:([cl].[id_cliente]=[open20].[dbo].[tickets].[id_cliente] as [t].[id_cliente]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
| |--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831
|--Clustered Index Seek(OBJECT:([open20].[dbo].[persona].[persona_id_persona] AS [per]), SEEK:([per].[id_persona]=[open20].[dbo].[tickets].[id_persona] as [t].[id_persona]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,000158	0,0032831

ANEXO 111

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 4 SELECT USANDO ÍNDICE + SUBQUERY 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	EstimateRows	EstimateIO	EstimateCPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1019]=[Expr1025], [Expr1023]=[open20].[dbo].[persona].[nombre] as [per].[nombre]))	Compute Scalar	Compute Scalar	1328688	0	0,132869	2371,892
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1328688	0	0,028517	2371,759
|--Hash Match(Right Outer Join, HASH:([per].[id_persona])=([t].[id_persona]))	Hash Match	Right Outer Join	1328688	0	2,849306	2371,73
|--Parallelism(Distribute Streams, Broadcast Partitioning)	Parallelism	Distribute Streams	10	0	0,028544	0,03183658
| |--Index Scan(OBJECT:([open20].[dbo].[persona].[persona_nombre] AS [per]))	Index Scan	Index Scan	10	0,003125	0,000168	0,003293
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([i].[id_ingreso], [l].[id_impuesto]))	Nested Loops	Left Outer Join	1328688	0	1,388479	2368,849
|--Compute Scalar(DEFINE:([Expr1015]=[open20].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	1328688	0	0,033217	497,6703
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso], [l].[id_impuesto]))	Parallelism	Repartition Streams	1328688	0	5,537490	497,6371
| |--Hash Match(Right Outer Join, HASH:([p].[id_producto])=([l].[id_producto]))	Hash Match	Right Outer Join	1328688	0	3,228439	492,0996
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([p].[id_producto]))	Parallelism	Repartition Streams	78763	0	0,158562	0,4493123
| | |--Index Scan(OBJECT:([open20].[dbo].[productos].[productos_nombre] AS [p]))	Index Scan	Index Scan	78763	0,2690509	0,021699	0,29075
| |--Compute Scalar(DEFINE:([Expr1011]=[open20].[dbo].[clientes].[nombres] as [cl].[nombres]))	Compute Scalar	Compute Scalar	1328688	0	0,033217	488,4218
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_producto]))	Parallelism	Repartition Streams	1328688	0	4,354127	488,3886
| |--Hash Match(Left Outer Join, HASH:([t].[id_cliente])=([cl].[id_cliente]), RESIDUAL:([open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente]=[open20].[dbo].[tickets].[id_cliente] as [t].[id_cliente]))	Hash Match	Left Outer Join	1328688	0	18,694470	484,0345
| |--Bitmap(HASH:([t].[id_cliente]), DEFINE:([Bitmap1036]))	Bitmap	Bitmap Create	1328688	0	3,208134	431,6266
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_cliente]))	Parallelism	Repartition Streams	1328688	0	3,208134	431,6266
| | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([t].[id_ticket]))	Hash Match	Inner Join	1328688	0	24,733980	428,4185
| | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Bitmap1035]))	Bitmap	Bitmap Create	132869	0	0,159443	332,7142
| | | |--Filter(WHERE:([open20].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso]=[open20].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso]))	Filter	Filter	132869	0	0,159443	332,7142
| | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	1328688	0	3,843015	332,5548
| | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Opt_Bitmap1027]))	Bitmap	Bitmap Create	442924	0	0,377884	1,111001
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	442924	0	0,377884	1,111001
| | | | |--Index Seek(OBJECT:([open20].[dbo].[pagos].[pagos_total] AS [pg]), SEEK:([pg].[total] >= (500) AND [pg].[total] <= (800)) ORDERED FORWARD)	Index Seek	Index Seek	442924	0,6112732	0,121844	0,7331166
| | | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([i].[id_ingreso]))	Hash Match	Inner Join	1328688	0	29,647140	327,6007
| | | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1034]))	Bitmap	Bitmap Create	1328688	0	1,832942	229,3976
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	1328688	0	1,832942	229,3976
| | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1027],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,498980	220,3651
| | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([i].[id_ingreso]))	Parallelism	Repartition Streams	20000000	0	17,184750	68,55606
| | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[ingresos].[ingresos_id_ingreso] AS [i]), WHERE:(PROBE([Bitmap1034],[open20].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87127	5,500039	51,37131
| | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	19,599750	70,97032
| | |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1035],[open20].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,500039	51,37057
| |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([cl].[id_cliente]))	Parallelism	Repartition Streams	2240818	0	3,728791	33,71336
| |--Clustered Index Scan(OBJECT:([open20].[dbo].[clientes].[clientes_id_cliente] AS [cl]), WHERE:(PROBE([Bitmap1036],[open20].[dbo].[clientes].[id_cliente] as [cl].[id_cliente],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	2240818	29,36831	0,616264	29,98458
|--Index Spool(SEEK:([i].[id_ingreso]=[open20].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso] AND [l].[id_impuesto]=[open20].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]))	Index Spool	Lazy Spool	1	0,003125	0,000258	1869,79
|--Assert(WHERE:(CASE WHEN [Expr1024]>(1) THEN (0) ELSE NULL END))	Assert	Assert	1	0	0,000000	1521,766
|--Stream Aggregate(DEFINE:([Expr1024]=Count(*), [Expr1025]=ANY([open20].[dbo].[lineasimpuesto].[monto] as [li].[monto])))	Stream Aggregate	Aggregate	1	0	0,000001	1521,554
|--Index Spool(SEEK:([li].[id_ingreso]=[open20].[dbo].[ingresos].[id_ingreso] as [i].[id_ingreso] AND [li].[id_impuesto]=[open20].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]))	Index Spool	Eager Spool	1	453,4861	20,000260	1521,026
|--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [li]))	Clustered Index Scan	Clustered Index Scan	20000000	53,10461	22,000160	75,10477

ANEXO 112

PLAN DE EJECUCIÓN EN SQL SERVER 2008 CONSULTA 4 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 20 MILLONES DE DATOS.

StmtText	PhysicalOp	LogicalOp	Estimate Rows	Estimate IO	Estimate CPU	TotalSubtree Cost
|--Compute Scalar(DEFINE:([Expr1021]=[open20].[dbo].[persona].[nombre] as [per].[nombre]))	Compute Scalar	Compute Scalar	1	0	0,0000001	529,2324
|--Parallelism(Gather Streams)	Parallelism	Gather Streams	1	0	0,0285166	529,2324
|--Nested Loops(Left Outer Join, OUTER REFERENCES:([t].[id_persona]))	Nested Loops	Left Outer Join	1	0	0,0000010	529,2039
|--Compute Scalar(DEFINE:([Expr1017]=[open20].[dbo].[productos].[nombre] as [p].[nombre]))	Compute Scalar	Compute Scalar	1	0	0,0000000	529,2006
| |--Nested Loops(Left Outer Join, OUTER REFERENCES:([l].[id_producto]))	Nested Loops	Left Outer Join	1	0	0,0000010	529,2006
| |--Compute Scalar(DEFINE:([Expr1013]=[open20].[dbo].[clientes].[nombres] as [cl].[nombres]))	Compute Scalar	Compute Scalar	1	0	0,0000000	529,1973
| | |--Nested Loops(Left Outer Join, OUTER REFERENCES:([t].[id_cliente]) OPTIMIZED)	Nested Loops	Left Outer Join	1	0	0,0000010	529,1973
| | |--Nested Loops(Inner Join, OUTER REFERENCES:([l].[id_ticket], [Expr1030]) WITH UNORDERED PREFETCH)	Nested Loops	Inner Join	1	0	0,1580075	529,194
| | | |--Sort(ORDER BY:([l].[id_ticket] ASC))	Sort	Sort	151203	0,002815315	2,9853890	454,977
| | | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso], [l].[id_impuesto])=([li].[id_ingreso], [li].[id_impuesto]), RESIDUAL:([open20].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso]=[open20].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso] AND [open20].[dbo].[lineasticket].[id_impuesto] as [l].[id_impuesto]=[open20].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto]))	Hash Match	Inner Join	151203	0	34,2037500	451,9888
| | | | |--Bitmap(HASH:([pg].[id_ingreso], [l].[id_impuesto]), DEFINE:([Bitmap1029]))	Bitmap	Bitmap Create	132869	0	0,1594426	338,3281
| | | | | |--Filter(WHERE:([open20].[dbo].[tickets].[id_ticket] as [t].[id_ticket]=[open20].[dbo].[pagos].[id_ingreso] as [pg].[id_ingreso]))	Filter	Filter	132869	0	0,1594426	338,3281
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso], [l].[id_impuesto]))	Parallelism	Repartition Streams	1328688	0	3,1996630	338,1687
| | | | | |--Hash Match(Inner Join, HASH:([pg].[id_ingreso])=([l].[id_ticket]))	Hash Match	Inner Join	1328688	0	3,8430150	334,969
| | | | | |--Bitmap(HASH:([pg].[id_ingreso]), DEFINE:([Opt_Bitmap1026]))	Bitmap	Bitmap Create	442924	0	0,3778843	1,111001
| | | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([pg].[id_ingreso]))	Parallelism	Repartition Streams	442924	0	0,3778843	1,111001
| | | | | | |--Index Seek(OBJECT:([open20].[dbo].[pagos].[pagos_total] AS [pg]), SEEK:([pg].[total] >= (500) AND [pg].[total] <= (800)) ORDERED FORWARD)	Index Seek	Index Seek	442924	0,6112732	0,1218435	0,7331166
| | | | | |--Hash Match(Inner Join, HASH:([l].[id_ticket])=([t].[id_ticket]))	Hash Match	Inner Join	1328688	0	29,6471400	330,015
| | | | | |--Bitmap(HASH:([l].[id_ticket]), DEFINE:([Bitmap1028]))	Bitmap	Bitmap Create	1328688	0	1,8329420	229,3976
| | | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([l].[id_ticket]))	Parallelism	Repartition Streams	1328688	0	1,8329420	229,3976
| | | | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasticket].[lineasticket_id_ticketlinea] AS [l]), WHERE:(PROBE([Opt_Bitmap1026],[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	59996160	203,8661	16,4989800	220,3651
| | | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([t].[id_ticket]))	Parallelism	Repartition Streams	20000000	0	19,5997500	70,97032
| | | | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[tickets].[tickets_id_ticket] AS [t]), WHERE:(PROBE([Bitmap1028],[open20].[dbo].[tickets].[id_ticket] as [t].[id_ticket],N'[IN ROW]')))	Clustered Index Scan	Clustered Index Scan	20000000	45,87053	5,5000390	51,37057
| | | | |--Parallelism(Repartition Streams, Hash Partitioning, PARTITION COLUMNS:([li].[id_ingreso], [li].[id_impuesto]))	Parallelism	Repartition Streams	20000000	0	20,8522500	79,45689
| | | | |--Clustered Index Scan(OBJECT:([open20].[dbo].[lineasimpuesto].[lineasimpuesto_id_lineasimpuestos] AS [li]), WHERE:(PROBE([Bitmap1029],[open20].[dbo].[lineasimpuesto].[id_ingreso] as [li].[id_ingreso],[open20].[dbo].[lineasimpuesto].[id_impuesto] as [li].[id_impuesto])))	Clustered Index Scan	Clustered Index Scan	20000000	53,10461	5,5000390	58,60464
| | | |--Clustered Index Seek(OBJECT:([open20].[dbo].[ingresos].[ingresos_id_ingreso] AS [i]), SEEK:([i].[id_ingreso]=[open20].[dbo].[lineasticket].[id_ticket] as [l].[id_ticket]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	74,05901
| |--Clustered Index Seek(OBJECT:([open20].[dbo].[clientes].[clientes_id_cliente] AS [cl]), SEEK:([cl].[id_cliente]=[open20].[dbo].[tickets].[id_cliente] as [t].[id_cliente]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831
|--Clustered Index Seek(OBJECT:([open20].[dbo].[productos].[productos_id_producto] AS [p]), SEEK:([p].[id_producto]=[open20].[dbo].[lineasticket].[id_producto] as [l].[id_producto]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831
|--Clustered Index Seek(OBJECT:([open20].[dbo].[persona].[persona_id_persona] AS [per]), SEEK:([per].[id_persona]=[open20].[dbo].[tickets].[id_persona] as [t].[id_persona]) ORDERED FORWARD)	Clustered Index Seek	Clustered Index Seek	1	0,003125	0,0001581	0,0032831

ANEXO 113

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 1 SELECT SIMPLE 10 MILLONES DE DATOS.

```
Plan hash value: 1639797780
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		856K	120M		133K (1)	00:26:45
* 1	HASH JOIN		856K	120M		133K (1)	00:26:45
2	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	1106	42028		5 (0)	00:00:01
* 3	HASH JOIN		856K	89M	3392K	133K (1)	00:26:44
4	TABLE ACCESS FULL	PRODUCTOS	78763	2461K		208 (1)	00:00:03
* 5	HASH JOIN		868K	63M	18M	129K (1)	00:25:56
* 6	HASH JOIN		289K	15M	15M	45919 (2)	00:09:12
* 7	HASH JOIN		286K	12M	23M	24486 (2)	00:04:54
* 8	TABLE ACCESS FULL	INGRESOS	860K	13M		8291 (2)	00:01:40
* 9	HASH JOIN		3333K	92M		8517 (2)	00:01:43
10	NESTED LOOPS		3	60		3 (0)	00:00:01
11	TABLE ACCESS BY INDEX ROWID	ROLES	1	14		1 (0)	00:00:01
* 12	INDEX UNIQUE SCAN	ROLES_NOMBRE	1			0 (0)	00:00:01
* 13	TABLE ACCESS FULL	PERSONA	3	18		2 (0)	00:00:01
14	TABLE ACCESS FULL	TICKETS	10M	419M		8469 (2)	00:01:42
* 15	TABLE ACCESS FULL	LINEASIMPUESTO	9965K	95M		10187 (2)	00:02:03
16	TABLE ACCESS FULL	LINEASTICKET	29M	600M		35701 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

1 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"="L"."ID_ATRIBUTOCONJUNTOINSTANCIA")
3 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
5 - access("T"."ID_TICKET"="L"."ID_TICKET")
6 - access("IM"."ID_INGRESO"="I"."ID_INGRESO")
7 - access("T"."ID_TICKET"="I"."ID_INGRESO")
8 - filter("I"."FECHA"<=TIMESTAMP' 2013-01-31 00:00:00' AND "I"."FECHA">=TIMESTAMP' 2013-01-01 00:00:00')
9 - access("T"."ID_PERSONA"="PS"."ID_PERSONA")
12 - access("R"."NOMBRE"='Dependiente')
13 - filter("PS"."ID_ROL"="R"."ID_ROL")
15 - filter("IM"."MONTO"<500)
```

ANEXO 114

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 1 SELECT USANDO ÍNDICE 10 MILLONES DE DATOS.

Plan hash value: 3434376623

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		841K	153M		736K (1)	02:27:14
* 1	HASH JOIN		841K	153M		736K (1)	02:27:14
2	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	1106	42028		5 (0)	00:00:01
* 3	HASH JOIN		841K	122M	3392K	736K (1)	02:27:14
4	TABLE ACCESS FULL	PRODUCTOS	78763	2461K		208 (1)	00:00:03
* 5	HASH JOIN		841K	97M	18M	730K (1)	02:26:06
* 6	HASH JOIN		289K	15M	15M	587K (1)	01:57:32
* 7	HASH JOIN		286K	12M	23M	566K (1)	01:53:15
8	TABLE ACCESS BY INDEX ROWID	INGRESOS	860K	13M		550K (1)	01:50:01
* 9	INDEX RANGE SCAN	INGRESOS_FECHA	860K			2289 (1)	00:00:28
* 10	HASH JOIN		3333K	92M		8517 (2)	00:01:43
11	NESTED LOOPS		3	60		3 (0)	00:00:01
12	TABLE ACCESS BY INDEX ROWID	ROLES	1	14		1 (0)	00:00:01
* 13	INDEX UNIQUE SCAN	ROLES_NOMBRE	1			0 (0)	00:00:01
* 14	TABLE ACCESS FULL	PERSONA	3	18		2 (0)	00:00:01
15	TABLE ACCESS FULL	TICKETS	10M	219M		8469 (2)	00:01:42
* 16	TABLE ACCESS FULL	LINEASIMPUESTO	9965K	95M		10187 (2)	00:02:03
17	TABLE ACCESS FULL	LINEASTICKET	29M	1803M		35690 (2)	00:07:09

Predicate Information (identified by operation id):

```
   1 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"="L"."ID_ATRIBUTOCONJUNTOINSTANCIA")
   3 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
   5 - access("T"."ID_TICKET"="L"."ID_TICKET")
   6 - access("IM"."ID_INGRESO"="I"."ID_INGRESO")
   7 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   9 - access("I"."FECHA">=TIMESTAMP' 2013-01-01 00:00:00' AND "I"."FECHA"<=TIMESTAMP' 2013-01-31 00:00:00')
  10 - access("T"."ID_PERSONA"="PS"."ID_PERSONA")
  13 - access("R"."NOMBRE"='Dependiente')
  14 - filter("PS"."ID_ROL"="R"."ID_ROL")
  16 - filter("IM"."MONTO"<500)
```

ANEXO 115

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 1 SELECT USANDO SUBQUERY 10 MILLONES DE DATOS.

Plan hash value: 3849619915

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		2626K	85M		75M (1)	252:33:34
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	ATRIBUTOCONJUNTOINSTANCIA	1	38		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	ATRIBUTOCONJUNTOINSTANCIA_ID_A	1			1 (0)	00:00:01
* 5	FILTER						
* 6	HASH JOIN RIGHT SEMI		2626K	85M	20M	113K (1)	00:22:48
7	VIEW	VW_NSO_1	868K	10M		30120 (2)	00:06:02
* 8	HASH JOIN		868K	22M	23M	30120 (2)	00:06:02
* 9	TABLE ACCESS FULL	INGRESOS	860K	13M		8291 (2)	00:01:40
* 10	TABLE ACCESS FULL	LINEASIMPUESTO	9965K	95M		10187 (2)	00:02:03
11	TABLE ACCESS FULL	LINEASTICKET	29M	600M		35728 (2)	00:07:09
12	NESTED LOOPS						
13	NESTED LOOPS		1	55		29 (0)	00:00:01
14	NESTED LOOPS		11	506		7 (0)	00:00:01
15	NESTED LOOPS		1	29		5 (0)	00:00:01
16	NESTED LOOPS		1	23		4 (0)	00:00:01
17	TABLE ACCESS BY INDEX ROWID	TICKETS	1	9		3 (0)	00:00:01
* 18	INDEX UNIQUE SCAN	TICKETS_ID_TICKET	1			2 (0)	00:00:01
19	TABLE ACCESS BY INDEX ROWID	ROLES	1	14		1 (0)	00:00:01
* 20	INDEX UNIQUE SCAN	ROLES_NOMBRE	1			0 (0)	00:00:01
* 21	TABLE ACCESS BY INDEX ROWID	PERSONA	10	60		1 (0)	00:00:01
* 22	INDEX UNIQUE SCAN	PERSONA_ID_PERSONA	1			0 (0)	00:00:01
* 23	TABLE ACCESS FULL	INGRESOS	11	187		2 (0)	00:00:01
* 24	INDEX UNIQUE SCAN	TICKETS_ID_TICKET	1			1 (0)	00:00:01
* 25	TABLE ACCESS BY INDEX ROWID	TICKETS	1	9		2 (0)	00:00:01

Predicate Information (identified by operation id):

```
   2 - access("P"."ID_PRODUCTO"=:B1)
   4 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"=:B1)
   5 - filter( EXISTS (SELECT 0 FROM "TICKETS" "TICKETS","PERSONA" "PERSONA","TICKETS" "TICKETS","INGRESOS"
              "INGRESOS","ROLES" "ROLES" WHERE "NOMBRE"='Dependiente' AND "FECHA"<=TIMESTAMP' 2013-01-31 00:00:00' AND
              "FECHA">=TIMESTAMP' 2013-01-01 00:00:00' AND "ID_TICKET"="ID_INGRESO" AND "ID_PERSONA"="ID_PERSONA" AND
              "ID_PERSONA"="ID_PERSONA" AND "ID_ROL"="ID_ROL" AND "ID_TICKET"=:B1))
   6 - access("ID_TICKET"="ID_INGRESO")
   8 - access("ID_INGRESO"="ID_INGRESO")
   9 - filter("FECHA"<=TIMESTAMP' 2013-01-31 00:00:00' AND "FECHA">=TIMESTAMP' 2013-01-01 00:00:00')
  10 - filter("MONTO"<500)
  18 - access("ID_TICKET"=:B1)
  20 - access("NOMBRE"='Dependiente')
  21 - filter("ID_ROL"="ID_ROL")
  22 - access("ID_PERSONA"="ID_PERSONA")
  23 - filter("FECHA"<=TIMESTAMP' 2013-01-31 00:00:00' AND "FECHA">=TIMESTAMP' 2013-01-01 00:00:00')
  24 - access("ID_TICKET"="ID_INGRESO")
  25 - filter("ID_PERSONA"="ID_PERSONA")
```

ANEXO 116

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 1 SELECT USANDO JOIN 10 MILLONES DE DATOS.

```
Plan hash value: 1639797780
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		840K	153M		194K (1)	00:38:53
* 1	HASH JOIN		840K	153M		194K (1)	00:38:53
2	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	1106	42028		5 (0)	00:00:01
* 3	HASH JOIN		841K	122M	3392K	194K (1)	00:38:53
4	TABLE ACCESS FULL	PRODUCTOS	78763	2461K		208 (1)	00:00:03
* 5	HASH JOIN		841K	97M	18M	188K (1)	00:37:45
* 6	HASH JOIN		289K	15M	15M	45919 (2)	00:09:12
* 7	HASH JOIN		286K	12M	23M	24486 (2)	00:04:54
* 8	TABLE ACCESS FULL	INGRESOS	860K	13M		8291 (2)	00:01:40
* 9	HASH JOIN		3333K	92M		8517 (2)	00:01:43
10	NESTED LOOPS		3	60		3 (0)	00:00:01
11	TABLE ACCESS BY INDEX ROWID	ROLES	1	14		1 (0)	00:00:01
* 12	INDEX UNIQUE SCAN	ROLES_NOMBRE	1			0 (0)	00:00:01
* 13	TABLE ACCESS FULL	PERSONA	3	18		2 (0)	00:00:01
14	TABLE ACCESS FULL	TICKETS	10M	839M		8469 (2)	00:01:42
* 15	TABLE ACCESS FULL	LINEASIMPUESTO	9965K	95M		10187 (2)	00:02:03
16	TABLE ACCESS FULL	LINEASTICKET	29M	1803M		35690 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"="L"."ID_ATRIBUTOCONJUNTOINSTANCIA")
   3 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
   5 - access("T"."ID_TICKET"="L"."ID_TICKET")
   6 - access("IM"."ID_INGRESO"="I"."ID_INGRESO")
   7 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   8 - filter("I"."FECHA"<=TIMESTAMP' 2013-01-31 00:00:00' AND "I"."FECHA">=TIMESTAMP' 2013-01-01 00:00:00')
   9 - access("PS"."ID_PERSONA"="T"."ID_PERSONA")
  12 - access("R"."NOMBRE"='Dependiente')
  13 - filter("PS"."ID_ROL"="R"."ID_ROL")
  15 - filter("IM"."MONTO"<500)
```

ANEXO 117

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 1 SELECT USANDO SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
Plan hash value: 2070554531
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		841K	97M		188K (1)	00:37:45
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	ATRIBUTOCONJUNTOINSTANCIA	1	38		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	ATRIBUTOCONJUNTOINSTANCIA_ID_A	1			1 (0)	00:00:01
* 5	HASH JOIN		841K	97M	18M	188K (1)	00:37:45
* 6	HASH JOIN		289K	15M	15M	45919 (2)	00:09:12
* 7	HASH JOIN		286K	12M	23M	24486 (2)	00:04:54
* 8	TABLE ACCESS FULL	INGRESOS	860K	13M		8291 (2)	00:01:40
* 9	HASH JOIN		3333K	92M		8517 (2)	00:01:43
10	NESTED LOOPS		3	60		3 (0)	00:00:01
11	TABLE ACCESS BY INDEX ROWID	ROLES	1	14		1 (0)	00:00:01
* 12	INDEX UNIQUE SCAN	ROLES_NOMBRE	1			0 (0)	00:00:01
* 13	TABLE ACCESS FULL	PERSONA	3	18		2 (0)	00:00:01
14	TABLE ACCESS FULL	TICKETS	10M	839M		8469 (2)	00:01:42
* 15	TABLE ACCESS FULL	LINEASIMPUESTO	9965K	95M		10187 (2)	00:02:03
16	TABLE ACCESS FULL	LINEASTICKET	29M	1803M		35690 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   4 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"=:B1)
   5 - access("T"."ID_TICKET"="L"."ID_TICKET")
   6 - access("IM"."ID_INGRESO"="I"."ID_INGRESO")
   7 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   8 - filter("I"."FECHA"<=TIMESTAMP' 2013-01-31 00:00:00' AND "I"."FECHA">=TIMESTAMP' 2013-01-01 00:00:00')
   9 - access("PS"."ID_PERSONA"="T"."ID_PERSONA")
  12 - access("R"."NOMBRE"='Dependiente')
  13 - filter("PS"."ID_ROL"="R"."ID_ROL")
  15 - filter("IM"."MONTO"<500)
```

ANEXO 118

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 1 SELECT USANDO ÍNDICE + JOIN 10 MILLONES DE DATOS..

```
Plan hash value: 3434376623
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		840K	153M		736K (1)	02:27:14
* 1	HASH JOIN		840K	153M		736K (1)	02:27:14
2	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	1106	42028		5 (0)	00:00:01
* 3	HASH JOIN		841K	122M	3392K	736K (1)	02:27:14
4	TABLE ACCESS FULL	PRODUCTOS	78763	2461K		208 (1)	00:00:03
* 5	HASH JOIN		841K	97M	18M	730K (1)	02:26:06
* 6	HASH JOIN		289K	15M	15M	587K (1)	01:57:32
* 7	HASH JOIN		286K	12M	23M	566K (1)	01:53:15
8	TABLE ACCESS BY INDEX ROWID	INGRESOS	860K	13M		550K (1)	01:50:01
* 9	INDEX RANGE SCAN	INGRESOS_FECHA	860K			2289 (1)	00:00:28
* 10	HASH JOIN		3333K	92M		8517 (2)	00:01:43
11	NESTED LOOPS		3	60		3 (0)	00:00:01
12	TABLE ACCESS BY INDEX ROWID	ROLES	1	14		1 (0)	00:00:01
* 13	INDEX UNIQUE SCAN	ROLES_NOMBRE	1			0 (0)	00:00:01
* 14	TABLE ACCESS FULL	PERSONA	3	18		2 (0)	00:00:01
15	TABLE ACCESS FULL	TICKETS	10M	219M		8469 (2)	00:01:42
* 16	TABLE ACCESS FULL	LINEASIMPUESTO	9965K	95M		10187 (2)	00:02:03
17	TABLE ACCESS FULL	LINEASTICKET	29M	1803M		35690 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"="L"."ID_ATRIBUTOCONJUNTOINSTANCIA")
   3 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
   5 - access("T"."ID_TICKET"="L"."ID_TICKET")
   6 - access("IM"."ID_INGRESO"="I"."ID_INGRESO")
   7 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   9 - access("I"."FECHA">=TIMESTAMP' 2013-01-01 00:00:00' AND "I"."FECHA"<=TIMESTAMP' 2013-01-31 00:00:00')
  10 - access("PS"."ID_PERSONA"="T"."ID_PERSONA")
  13 - access("R"."NOMBRE"='Dependiente')
  14 - filter("PS"."ID_ROL"="R"."ID_ROL")
  16 - filter("IM"."MONTO"<500)
```

ANEXO 119

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 1 SELECT CON ÍNDICE + SUBQUERY 10 MILLONES DE DATOS.

Plan hash value: 3477081312

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		29M	2524M		344M (2)	999:59:59
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	ATRIBUTOCONJUNTOINSTANCIA	1	38		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	ATRIBUTOCONJUNTOINSTANCIA_ID_A	1			1 (0)	00:00:01
* 5	HASH JOIN SEMI		29M	2524M	2496M	344M (2)	999:59:59
* 6	HASH JOIN RIGHT SEMI		29M	2163M	20M	714K (1)	02:22:57
7	VIEW	VW_NSO_1	868K	10M		571K (1)	01:54:23
* 8	HASH JOIN		868K	22M	23M	571K (1)	01:54:23
9	TABLE ACCESS BY INDEX ROWID	INGRESOS	860K	13M		550K (1)	01:50:01
* 10	INDEX RANGE SCAN	INGRESOS_FECHA	860K			2289 (1)	00:00:28
* 11	TABLE ACCESS FULL	LINEASIMPUESTO	9965K	95M		10187 (2)	00:02:03
12	TABLE ACCESS FULL	LINEASTICKET	29M	1803M		35690 (2)	00:07:09
13	VIEW	VW_NSO_2	286G	3473G		1880K (69)	06:16:10
* 14	HASH JOIN		286G	14T	15M	1880K (69)	06:16:10
* 15	HASH JOIN		286K	12M	23M	566K (1)	01:53:15
16	TABLE ACCESS BY INDEX ROWID	INGRESOS	860K	13M		550K (1)	01:50:01
* 17	INDEX RANGE SCAN	INGRESOS_FECHA	860K			2289 (1)	00:00:28
* 18	HASH JOIN		3333K	92M		8517 (2)	00:01:43
19	NESTED LOOPS		3	60		3 (0)	00:00:01
20	TABLE ACCESS BY INDEX ROWID	ROLES	1	14		1 (0)	00:00:01
* 21	INDEX UNIQUE SCAN	ROLES_NOMBRE	1			0 (0)	00:00:01
* 22	TABLE ACCESS FULL	PERSONA	3	18		2 (0)	00:00:01
23	TABLE ACCESS FULL	TICKETS	10M	85M		8469 (2)	00:01:42
24	TABLE ACCESS FULL	TICKETS	10M	85M		8469 (2)	00:01:42

Predicate Information (identified by operation id):

```
   2 - access("P"."ID_PRODUCTO"=:B1)
   4 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"=:B1)
   5 - access("ID_TICKET"="ID_TICKET")
   6 - access("ID_TICKET"="ID_INGRESO")
   8 - access("ID_INGRESO"="ID_INGRESO")
  10 - access("FECHA">=TIMESTAMP' 2013-01-01 00:00:00' AND "FECHA"<=TIMESTAMP' 2013-01-31 00:00:00')
  11 - filter("MONTO"<500)
  14 - access("ID_PERSONA"="ID_PERSONA")
  15 - access("ID_TICKET"="ID_INGRESO")
  17 - access("FECHA">=TIMESTAMP' 2013-01-01 00:00:00' AND "FECHA"<=TIMESTAMP' 2013-01-31 00:00:00')
  18 - access("ID_PERSONA"="ID_PERSONA")
  21 - access("NOMBRE"='Dependiente')
  22 - filter("ID_ROL"="ID_ROL")
```

ANEXO 120

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 1 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 10 MILLONES DE DATOS.

Plan hash value: 2538894714

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		841K	97M		730K (1)	02:26:06
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	ATRIBUTOCONJUNTOINSTANCIA	1	38		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	ATRIBUTOCONJUNTOINSTANCIA_ID_A	1			1 (0)	00:00:01
* 5	HASH JOIN		841K	97M	18M	730K (1)	02:26:06
* 6	HASH JOIN		289K	15M	15M	587K (1)	01:57:32
* 7	HASH JOIN		286K	12M	23M	566K (1)	01:53:15
8	TABLE ACCESS BY INDEX ROWID	INGRESOS	860K	13M		550K (1)	01:50:01
* 9	INDEX RANGE SCAN	INGRESOS_FECHA	860K			2289 (1)	00:00:28
* 10	HASH JOIN		3333K	92M		8517 (2)	00:01:43
11	NESTED LOOPS		3	60		3 (0)	00:00:01
12	TABLE ACCESS BY INDEX ROWID	ROLES	1	14		1 (0)	00:00:01
* 13	INDEX UNIQUE SCAN	ROLES_NOMBRE	1			0 (0)	00:00:01
* 14	TABLE ACCESS FULL	PERSONA	3	18		2 (0)	00:00:01
15	TABLE ACCESS FULL	TICKETS	10M	839M		8469 (2)	00:01:42
* 16	TABLE ACCESS FULL	LINEASIMPUESTO	9965K	95M		10187 (2)	00:02:03
17	TABLE ACCESS FULL	LINEASTICKET	29M	1803M		35690 (2)	00:07:09

Predicate Information (identified by operation id):

```
   2 - access("P"."ID_PRODUCTO"=:B1)
   4 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"=:B1)
   5 - access("T"."ID_TICKET"="L"."ID_TICKET")
   6 - access("IM"."ID_INGRESO"="I"."ID_INGRESO")
   7 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   9 - access("I"."FECHA">=TIMESTAMP' 2013-01-01 00:00:00' AND "I"."FECHA"<=TIMESTAMP' 2013-01-31 00:00:00')
  10 - access("PS"."ID_PERSONA"="T"."ID_PERSONA")
  13 - access("R"."NOMBRE"='Dependiente')
  14 - filter("PS"."ID_ROL"="R"."ID_ROL")
  16 - filter("IM"."MONTO"<500)
```

ANEXO 121

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 2 SELECT SIMPLE Y SELECT USANDO JOIN 10 MILLONES DE DATOS.

Plan hash value: 2879677815

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		67506	10M		74184 (2)	00:14:51
* 1	HASH JOIN		67506	10M	8704K	74184 (2)	00:14:51
* 2	HASH JOIN		67506	7910K	7912K	57042 (2)	00:11:25
* 3	HASH JOIN		67506	7119K	3848K	36743 (2)	00:07:21
4	TABLE ACCESS FULL	PRODUCTOS	78763	2922K		208 (1)	00:00:03
* 5	HASH JOIN		136K	9355K		35815 (2)	00:07:10
6	NESTED LOOPS		5	260		6 (0)	00:00:01
7	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	1	14		1 (0)	00:00:01
* 8	INDEX UNIQUE SCAN	CATEGORIAS_NOMBRE	1			0 (0)	00:00:01
* 9	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	5	190		5 (0)	00:00:01
10	TABLE ACCESS FULL	LINEASTICKET	29M	514M		35673 (1)	00:07:09
11	TABLE ACCESS FULL	TICKETS	10M	114M		8478 (2)	00:01:42
12	TABLE ACCESS FULL	CLIENTES	2240K	102M		10343 (1)	00:02:05

Predicate Information (identified by operation id):

```
   1 - access("CL"."ID_CLIENTE"="T"."ID_CLIENTE")
   2 - access("T"."ID_TICKET"="L"."ID_TICKET")
   3 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO" AND "C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   5 - access("ACI"."ID_ATRIBUTOCONJUNTOINSTANCIA"="L"."ID_ATRIBUTOCONJUNTOINSTANCIA")
   8 - access("C"."NOMBRE"='FARMACEUTICOS')
   9 - filter("ACI"."DESCRIPCION"='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR
              "ACI"."DESCRIPCION"='ATRIB.CONJUNTO NATURPHARMA' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX GENERICOS'
              OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO VITAFARMA')
```

ANEXO 122

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 2 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 10 MILLONES DE DATOS.

```
Plan hash value: 1758223253
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		67506	10M		161K (1)	00:32:15
* 1	HASH JOIN		67506	10M	8704K	161K (1)	00:32:15
* 2	HASH JOIN		67506	7910K	7912K	144K (1)	00:28:49
* 3	HASH JOIN		67506	7119K	3848K	123K (1)	00:24:45
4	TABLE ACCESS FULL	PRODUCTOS	78763	2922K		208 (1)	00:00:03
5	NESTED LOOPS						
6	NESTED LOOPS		136K	9355K		122K (1)	00:24:34
7	NESTED LOOPS		5	260		6 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	1	14		1 (0)	00:00:01
* 9	INDEX UNIQUE SCAN	CATEGORIAS_NOMBRE	1			0 (0)	00:00:01
* 10	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	5	190		5 (0)	00:00:01
* 11	INDEX RANGE SCAN	ATRIBCONJINST_DESCRIP	27124			57 (0)	00:00:01
12	TABLE ACCESS BY INDEX ROWID	LINEASTICKET	27124	476K		24562 (1)	00:04:55
13	TABLE ACCESS FULL	TICKETS	10M	114M		8478 (2)	00:01:42
14	TABLE ACCESS FULL	CLIENTES	2240K	102M		10343 (1)	00:02:05

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("CL"."ID_CLIENTE"="T"."ID_CLIENTE")
   2 - access("T"."ID_TICKET"="L"."ID_TICKET")
   3 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO" AND "C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   9 - access("C"."NOMBRE"='FARMACEUTICOS')
  10 - filter("ACI"."DESCRIPCION"='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO
              NATURPHARMA' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO
              QUIFATEX PROP MERZ' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO VITAFARMA')
  11 - access("ACI"."ID_ATRIBUTOCONJUNTOINSTANCIA"="L"."ID_ATRIBUTOCONJUNTOINSTANCIA")
```

ANEXO 123

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 2 SELECT USANDO SUBQUERY 10 MILLONES DE DATOS.

```
Plan hash value: 3539947530
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		132K	18M		73862 (2)	00:14:47
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	NESTED LOOPS		1	22		3 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 9	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
10	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 11	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
12	TABLE ACCESS BY INDEX ROWID	ATRIBUTOCONJUNTOINSTANCIA	1	38		2 (0)	00:00:01
* 13	INDEX UNIQUE SCAN	ATRIBUTOCONJUNTOINSTANCIA_ID_A	1			1 (0)	00:00:01
* 14	FILTER						
* 15	HASH JOIN		132K	18M	14M	73859 (2)	00:14:47
* 16	HASH JOIN		132K	12M	12M	56423 (2)	00:11:18
* 17	HASH JOIN		132K	11M		35866 (2)	00:07:11
* 18	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	5	190		5 (0)	00:00:01
19	TABLE ACCESS FULL	LINEASTICKET	29M	1442M		35730 (2)	00:07:09
20	TABLE ACCESS FULL	TICKETS	10M	114M		8478 (2)	00:01:42
21	TABLE ACCESS FULL	CLIENTES	2240K	102M		10343 (1)	00:02:05

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   5 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   9 - access("P"."ID_PRODUCTO"=:B1)
  11 - access("P"."ID_PRODUCTO"=:B1)
  13 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"=:B1)
  14 - filter( (SELECT "C"."NOMBRE" FROM "CATEGORIAS" "C","PRODUCTOS" "P" WHERE "P"."ID_PRODUCTO"=:B1 AND
              "C"."ID_CATEGORIA"="P"."ID_CATEGORIA")='FARMACEUTICOS')
  15 - access("T"."ID_CLIENTE"="CL"."ID_CLIENTE")
  16 - access("L"."ID_TICKET"="T"."ID_TICKET")
  17 - access("L"."ID_ATRIBUTOCONJUNTOINSTANCIA"="A"."ID_ATRIBUTOCONJUNTOINSTANCIA")
  18 - filter("DESCRIPCION"='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR "DESCRIPCION"='ATRIB.CONJUNTO
              NATURPHARMA' OR "DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR "DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX
              PROP MERZ' OR "DESCRIPCION"='ATRIB.CONJUNTO VITAFARMA')
```

ANEXO 124

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 2 SELECT USANDO SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
Plan hash value: 2036587207
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		136K	45M		73482 (2)	00:14:42
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	NESTED LOOPS		1	22		3 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 9	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
10	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 11	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
12	VIEW		136K	45M		73482 (2)	00:14:42
* 13	FILTER						
* 14	HASH JOIN		136K	15M	10M	73479 (2)	00:14:42
* 15	HASH JOIN		136K	9088K	9096K	56239 (2)	00:11:15
* 16	HASH JOIN		136K	7484K		35882 (2)	00:07:11
* 17	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	5	190		5 (0)	00:00:01
18	TABLE ACCESS FULL	LINEASTICKET	29M	514M		35741 (2)	00:07:09
19	TABLE ACCESS FULL	TICKETS	10M	114M		8478 (2)	00:01:42
20	TABLE ACCESS FULL	CLIENTES	2240K	102M		10343 (1)	00:02:05

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   5 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   9 - access("P"."ID_PRODUCTO"=:B1)
  11 - access("P"."ID_PRODUCTO"=:B1)
  13 - filter( (SELECT "C"."NOMBRE" FROM "CATEGORIAS" "C","PRODUCTOS" "P" WHERE "P"."ID_PRODUCTO"=:B1 AND
              "C"."ID_CATEGORIA"="P"."ID_CATEGORIA")='FARMACEUTICOS')
  14 - access("T"."ID_CLIENTE"="CL"."ID_CLIENTE")
  15 - access("L"."ID_TICKET"="T"."ID_TICKET")
  16 - access("ACI"."ID_ATRIBUTOCONJUNTOINSTANCIA"="L"."ID_ATRIBUTOCONJUNTOINSTANCIA")
  17 - filter("ACI"."DESCRIPCION"='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR
              "ACI"."DESCRIPCION"='ATRIB.CONJUNTO NATURPHARMA' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX
              GENERICOS' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR
              "ACI"."DESCRIPCION"='ATRIB.CONJUNTO VITAFARMA')
```

ANEXO 125

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 2 SELECT CON ÍNDICE + SUBQUERY 10 MILLONES DE DATOS.

```
Plan hash value: 746685256
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		132K	18M		38696 (1)	00:07:45
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	NESTED LOOPS		1	22		3 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 9	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
10	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 11	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
12	TABLE ACCESS BY INDEX ROWID	ATRIBUTOCONJUNTOINSTANCIA	1	38		2 (0)	00:00:01
* 13	INDEX UNIQUE SCAN	ATRIBUTOCONJUNTOINSTANCIA_ID_A	1			1 (0)	00:00:01
* 14	FILTER						
* 15	HASH JOIN		132K	18M	14M	38693 (1)	00:07:45
* 16	HASH JOIN		132K	12M	12M	21256 (1)	00:04:16
17	NESTED LOOPS						
18	NESTED LOOPS		132K	11M		700 (0)	00:00:09
* 19	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	5	190		5 (0)	00:00:01
* 20	INDEX RANGE SCAN	ATRIBCONJINST_DESCRIP	87			57 (0)	00:00:01
21	TABLE ACCESS BY INDEX ROWID	LINEASTICKET	26301	1335K		139 (0)	00:00:02
22	TABLE ACCESS FULL	TICKETS	10M	114M		8478 (2)	00:01:42
23	TABLE ACCESS FULL	CLIENTES	2240K	102M		10343 (1)	00:02:05

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   5 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   9 - access("P"."ID_PRODUCTO"=:B1)
  11 - access("P"."ID_PRODUCTO"=:B1)
  13 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"=:B1)
  14 - filter( (SELECT "C"."NOMBRE" FROM "CATEGORIAS" "C","PRODUCTOS" "P" WHERE "P"."ID_PRODUCTO"=:B1 AND
              "C"."ID_CATEGORIA"="P"."ID_CATEGORIA")='FARMACEUTICOS')
  15 - access("T"."ID_CLIENTE"="CL"."ID_CLIENTE")
  16 - access("L"."ID_TICKET"="T"."ID_TICKET")
  19 - filter("DESCRIPCION"='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR "DESCRIPCION"='ATRIB.CONJUNTO NATURPHARMA'
             OR "DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR "DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR
             "DESCRIPCION"='ATRIB.CONJUNTO VITAFARMA')
  20 - access("L"."ID_ATRIBUTOCONJUNTOINSTANCIA"="A"."ID_ATRIBUTOCONJUNTOINSTANCIA")
```

ANEXO 126

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 2 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
Plan hash value: 821077450
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		132K	44M		160K (1)	00:32:09
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	NESTED LOOPS		1	22		3 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 9	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
10	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 11	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
12	VIEW		132K	44M		160K (1)	00:32:09
* 13	FILTER						
* 14	HASH JOIN		132K	18M	14M	160K (1)	00:32:09
* 15	HASH JOIN		132K	12M	12M	143K (1)	00:28:40
16	NESTED LOOPS						
17	NESTED LOOPS		132K	11M		122K (1)	00:24:33
* 18	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	5	190		5 (0)	00:00:01
* 19	INDEX RANGE SCAN	ATRIBCONJINST_DESCRIP	26277			57 (0)	00:00:01
20	TABLE ACCESS BY INDEX ROWID	LINEASTICKET	26277	1334K		24540 (1)	00:04:55
21	TABLE ACCESS FULL	TICKETS	10M	114M		8478 (2)	00:01:42
22	TABLE ACCESS FULL	CLIENTES	2240K	102M		10343 (1)	00:02:05

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   5 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   9 - access("P"."ID_PRODUCTO"=:B1)
  11 - access("P"."ID_PRODUCTO"=:B1)
  13 - filter( (SELECT "C"."NOMBRE" FROM "CATEGORIAS" "C","PRODUCTOS" "P" WHERE "P"."ID_PRODUCTO"=:B1 AND
              "C"."ID_CATEGORIA"="P"."ID_CATEGORIA")='FARMACEUTICOS')
  14 - access("T"."ID_CLIENTE"="CL"."ID_CLIENTE")
  15 - access("L"."ID_TICKET"="T"."ID_TICKET")
  18 - filter("ACI"."DESCRIPCION"='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR
              "ACI"."DESCRIPCION"='ATRIB.CONJUNTO NATURPHARMA' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX GENERICOS'
              OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO VITAFARMA')
  19 - access("ACI"."ID_ATRIBUTOCONJUNTOINSTANCIA"="L"."ID_ATRIBUTOCONJUNTOINSTANCIA")
```

ANEXO 127

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 3 SELECT SIMPLE Y SELECT USANDO JOIN 10 MILLONES DE DATOS.

```
Plan hash value: 4030141452
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		25445	2211K		209K (1)	00:42:00
*1	FILTER						
2	HASH GROUP BY		25445	2211K	984M	209K (1)	00:42:00
*3	HASH JOIN		10M	863M		136K (1)	00:27:14
4	TABLE ACCESS FULL	CATEGORIAS	2	28		2 (0)	00:00:01
*5	HASH JOIN		10M	727M	3616K	136K (1)	00:27:14
6	TABLE ACCESS FULL	PRODUCTOS	78763	2692K		208 (1)	00:00:03
*7	HASH JOIN		10M	393M	114M	110K (1)	00:22:04
*8	HASH JOIN		3439K	75M	95M	27452 (2)	00:05:30
*9	TABLE ACCESS FULL	INGRESOS	3439K	55M		8304 (2)	00:01:40
10	INDEX FAST FULL SCAN	TICKETS_ID_TICKET	10M	57M		5809 (2)	00:01:10
11	TABLE ACCESS FULL	LINEASTICKET	29M	486M		35701 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(SUM("L"."UNIDADES")>=600 AND SUM("L"."UNIDADES")<=1000)
   3 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   5 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
   7 - access("T"."ID_TICKET"="L"."ID_TICKET")
   8 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   9 - filter("I"."FECHA">=TIMESTAMP' 2013-05-01 00:00:00' AND "I"."FECHA"<=TIMESTAMP'
              2013-08-31 00:00:00')
```

ANEXO 128

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 3 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 10 MILLONES DE DATOS.

```
Plan hash value: 3365292073
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		394	48856		2388K (1)	07:57:41
* 1	FILTER						
2	HASH GROUP BY		394	48856		2388K (1)	07:57:41
* 3	HASH JOIN		10M	1183M		2387K (1)	07:57:36
4	TABLE ACCESS FULL	CATEGORIAS	2	28		2 (0)	00:00:01
* 5	HASH JOIN		10M	1049M	3616K	2387K (1)	07:57:35
6	TABLE ACCESS FULL	PRODUCTOS	78763	2692K		208 (1)	00:00:03
* 7	HASH JOIN		10M	715M	114M	2346K (1)	07:49:15
* 8	HASH JOIN		3439K	75M	95M	2216K (1)	07:23:19
9	TABLE ACCESS BY INDEX ROWID	INGRESOS	3439K	55M		2197K (1)	07:19:30
* 10	INDEX RANGE SCAN	INGRESOS_FECHA	3439K			9135 (1)	00:01:50
11	INDEX FAST FULL SCAN	TICKETS_ID_TICKET	10M	57M		5809 (2)	00:01:10
12	TABLE ACCESS FULL	LINEASTICKET	29M	1442M		35690 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(SUM("L"."UNIDADES")>=600 AND SUM("L"."UNIDADES")<=1000)
   3 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   5 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
   7 - access("T"."ID_TICKET"="L"."ID_TICKET")
   8 - access("T"."ID_TICKET"="I"."ID_INGRESO")
  10 - access("I"."FECHA">=TIMESTAMP' 2013-05-01 00:00:00' AND "I"."FECHA"<=TIMESTAMP' 2013-08-31
              00:00:00')
```

ANEXO 129

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 3 SELECT USANDO SUBQUERY 10 MILLONES DE DATOS.

```
Plan hash value: 3356598634
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		29M	1803M		156K (2)	00:31:24
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	NESTED LOOPS		1	22		3 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 9	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
* 10	FILTER						
11	HASH GROUP BY		29M	1803M		156K (2)	00:31:24
* 12	HASH JOIN RIGHT SEMI		29M	1803M	82M	155K (1)	00:31:07
13	VIEW	VW_NSO_1	3439K	42M		27452 (2)	00:05:30
* 14	HASH JOIN		3439K	75M	95M	27452 (2)	00:05:30
* 15	TABLE ACCESS FULL	INGRESOS	3439K	55M		8304 (2)	00:01:40
16	INDEX FAST FULL SCAN	TICKETS_ID_TICKET	10M	57M		5809 (2)	00:01:10
17	TABLE ACCESS FULL	LINEASTICKET	29M	1442M		35690 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   5 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   9 - access("P"."ID_PRODUCTO"=:B1)
  10 - filter(SUM("L"."UNIDADES")>=600 AND SUM("L"."UNIDADES")<=1000)
  12 - access("L"."ID_TICKET"="ID_TICKET")
  14 - access("ID_TICKET"="ID_INGRESO")
  15 - filter("FECHA">=TIMESTAMP' 2013-05-01 00:00:00' AND "FECHA"<=TIMESTAMP' 2013-08-31 00:00:00')
```

ANEXO 130

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 3 SELECT USANDO SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
Plan hash value: 3504110572
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		140	14980		199K (1)	00:39:51
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	NESTED LOOPS		1	22		3 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
* 8	FILTER						
9	HASH GROUP BY		140	14980		199K (1)	00:39:51
* 10	HASH JOIN		10M	1020M	3392K	198K (1)	00:39:45
11	TABLE ACCESS FULL	PRODUCTOS	78763	2461K		208 (1)	00:00:03
* 12	HASH JOIN		10M	715M	114M	157K (1)	00:31:26
* 13	HASH JOIN		3439K	75M	95M	27452 (2)	00:05:30
* 14	TABLE ACCESS FULL	INGRESOS	3439K	55M		8304 (2)	00:01:40
15	INDEX FAST FULL SCAN	TICKETS_ID_TICKET	10M	57M		5809 (2)	00:01:10
16	TABLE ACCESS FULL	LINEASTICKET	29M	1442M		35690 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   5 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   8 - filter(SUM("L"."UNIDADES")>=600 AND SUM("L"."UNIDADES")<=1000)
  10 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
  12 - access("L"."ID_TICKET"="T"."ID_TICKET")
  13 - access("T"."ID_TICKET"="I"."ID_INGRESO")
  14 - filter("I"."FECHA">=TIMESTAMP' 2013-05-01 00:00:00' AND "I"."FECHA"<=TIMESTAMP' 2013-08-31
              00:00:00')
```

ANEXO 131

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 3 SELECT CON ÍNDICE + SUBQUERY 10 MILLONES DE DATOS.

```
Plan hash value: 87719989
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		29M	1803M		2346K (1)	07:49:14
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
5	NESTED LOOPS		1	22		3 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 9	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
* 10	FILTER						
11	HASH GROUP BY		29M	1803M		2346K (1)	07:49:14
* 12	HASH JOIN RIGHT SEMI		29M	1803M	82M	2344K (1)	07:48:56
13	VIEW	VW_NSO_1	3439K	42M		2216K (1)	07:23:19
* 14	HASH JOIN		3439K	75M	95M	2216K (1)	07:23:19
15	TABLE ACCESS BY INDEX ROWID	INGRESOS	3439K	55M		2197K (1)	07:19:30
* 16	INDEX RANGE SCAN	INGRESOS_FECHA	3439K			9135 (1)	00:01:50
17	INDEX FAST FULL SCAN	TICKETS_ID_TICKET	10M	57M		5809 (2)	00:01:10
18	TABLE ACCESS FULL	LINEASTICKET	29M	1442M		35690 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   4 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("P"."ID_PRODUCTO"=:B1)
   9 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
  10 - filter(SUM("L"."UNIDADES")>=600 AND SUM("L"."UNIDADES")<=1000)
  12 - access("L"."ID_TICKET"="ID_TICKET")
  14 - access("ID_TICKET"="ID_INGRESO")
  16 - access("FECHA">=TIMESTAMP' 2013-05-01 00:00:00' AND "FECHA"<=TIMESTAMP' 2013-08-31 00:00:00')
```

ANEXO 132

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 3 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
Plan hash value: 3889954275
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		140	14980		2388K (1)	07:57:40
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	NESTED LOOPS		1	22		3 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
* 8	FILTER						
9	HASH GROUP BY		140	14980		2388K (1)	07:57:40
* 10	HASH JOIN		10M	1020M	3392K	2387K (1)	07:57:35
11	TABLE ACCESS FULL	PRODUCTOS	78763	2461K		208 (1)	00:00:03
* 12	HASH JOIN		10M	715M	114M	2346K (1)	07:49:15
* 13	HASH JOIN		3439K	75M	95M	2216K (1)	07:23:19
14	TABLE ACCESS BY INDEX ROWID	INGRESOS	3439K	55M		2197K (1)	07:19:30
* 15	INDEX RANGE SCAN	INGRESOS_FECHA	3439K			9135 (1)	00:01:50
16	INDEX FAST FULL SCAN	TICKETS_ID_TICKET	10M	57M		5809 (2)	00:01:10
17	TABLE ACCESS FULL	LINEASTICKET	29M	1442M		35690 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   5 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   8 - filter(SUM("L"."UNIDADES")>=600 AND SUM("L"."UNIDADES")<=1000)
  10 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
  12 - access("L"."ID_TICKET"="T"."ID_TICKET")
  13 - access("T"."ID_TICKET"="I"."ID_INGRESO")
  15 - access("FECHA">=TIMESTAMP' 2013-05-01 00:00:00' AND "FECHA"<=TIMESTAMP' 2013-08-31 00:00:00')
```

ANEXO 133

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 4 SELECT SIMPLE Y SELECT USANDO JOIN 10 MILLONES DE DATOS.

```
Plan hash value: 4211421244
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		332K	54M		175K (1)	00:35:04
* 1	HASH JOIN		332K	54M		175K (1)	00:35:04
2	TABLE ACCESS FULL	PERSONA	10	310		2 (0)	00:00:01
* 3	HASH JOIN		332K	44M	3392K	175K (1)	00:35:04
4	TABLE ACCESS FULL	PRODUCTOS	78763	2461K		208 (1)	00:00:03
* 5	HASH JOIN		337K	35M	10M	172K (1)	00:34:36
* 6	HASH JOIN		112K	9774K	7472K	91077 (1)	00:18:13
* 7	HASH JOIN		112K	6150K	5824K	75583 (1)	00:15:08
* 8	HASH JOIN		112K	4502K	4400K	53967 (1)	00:10:48
* 9	HASH JOIN		112K	3075K	2528K	31678 (2)	00:06:21
* 10	TABLE ACCESS FULL	PAGOS	112K	1208K		9486 (2)	00:01:54
11	TABLE ACCESS FULL	INGRESOS	10M	267M		8266 (2)	00:01:40
12	TABLE ACCESS FULL	LINEASIMPUESTO	10M	123M		10165 (2)	00:02:02
13	TABLE ACCESS FULL	TICKETS	10M	410M		8478 (2)	00:01:42
14	TABLE ACCESS FULL	CLIENTES	2240K	70M		10343 (1)	00:02:05
15	TABLE ACCESS FULL	LINEASTICKET	29M	572M		35728 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("T"."ID_PERSONA"="PER"."ID_PERSONA")
   3 - access("L"."ID_PRODUCTO"="P"."ID_PRODUCTO")
   5 - access("T"."ID_TICKET"="L"."ID_TICKET" AND "L"."ID_IMPUESTO"="LI"."ID_IMPUESTO")
   6 - access("CL"."ID_CLIENTE"="T"."ID_CLIENTE")
   7 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   8 - access("LI"."ID_INGRESO"="I"."ID_INGRESO")
   9 - access("PG"."ID_INGRESO"="I"."ID_INGRESO")
  10 - filter("PG"."TOTAL"<=800 AND "PG"."TOTAL">=500)
```

ANEXO 134

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 4 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 10 MILLONES DE DATOS.

```
Plan hash value: 2144962465
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		332K	54M		229K (1)	00:45:50
*1	HASH JOIN		332K	54M		229K (1)	00:45:50
2	TABLE ACCESS FULL	PERSONA	10	310		2 (0)	00:00:01
*3	HASH JOIN		332K	44M	3392K	229K (1)	00:45:49
4	TABLE ACCESS FULL	PRODUCTOS	78763	2461K		208 (1)	00:00:03
*5	HASH JOIN		337K	35M	10M	226K (1)	00:45:22
*6	HASH JOIN		112K	9774K	7472K	144K (1)	00:28:59
*7	HASH JOIN		112K	6150K	5824K	129K (1)	00:25:53
*8	HASH JOIN		112K	4502K	4400K	107K (1)	00:21:33
*9	HASH JOIN		112K	3075K	2528K	85453 (1)	00:17:06
10	TABLE ACCESS BY INDEX ROWID	PAGOS	112K	1208K		63261 (1)	00:12:40
*11	INDEX RANGE SCAN	PAGOS_TOTAL	112K			243 (1)	00:00:03
12	TABLE ACCESS FULL	INGRESOS	10M	267M		8266 (2)	00:01:40
13	TABLE ACCESS FULL	LINEASIMPUESTO	10M	123M		10165 (2)	00:02:02
14	TABLE ACCESS FULL	TICKETS	10M	410M		8478 (2)	00:01:42
15	TABLE ACCESS FULL	CLIENTES	2240K	70M		10343 (1)	00:02:05
16	TABLE ACCESS FULL	LINEASTICKET	29M	572M		35728 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("T"."ID_PERSONA"="PER"."ID_PERSONA")
   3 - access("L"."ID_PRODUCTO"="P"."ID_PRODUCTO")
   5 - access("T"."ID_TICKET"="L"."ID_TICKET" AND "L"."ID_IMPUESTO"="LI"."ID_IMPUESTO")
   6 - access("CL"."ID_CLIENTE"="T"."ID_CLIENTE")
   7 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   8 - access("LI"."ID_INGRESO"="I"."ID_INGRESO")
   9 - access("PG"."ID_INGRESO"="I"."ID_INGRESO")
  11 - access("PG"."TOTAL">=500 AND "PG"."TOTAL"<=800)
```

ANEXO 135

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 4 SELECT USANDO SUBQUERY 10 MILLONES DE DATOS.

```
Plan hash value: 2112869973
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		337K	20M		134K (1)	00:26:59
1	TABLE ACCESS BY INDEX ROWID	CLIENTES	1	33		3 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	CLIENTES_ID_CLIENTE	1			2 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
* 5	TABLE ACCESS FULL	LINEASIMPUESTO	1	13		10169 (2)	00:02:03
6	TABLE ACCESS BY INDEX ROWID	PERSONA	1	31		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	PERSONA_ID_PERSONA	1			0 (0)	00:00:01
* 8	HASH JOIN		337K	20M	6048K	134K (1)	00:26:59
* 9	HASH JOIN		112K	4722K	4400K	53225 (1)	00:10:39
* 10	HASH JOIN		112K	3075K	2528K	31678 (2)	00:06:21
* 11	TABLE ACCESS FULL	PAGOS	112K	1208K		9486 (2)	00:01:54
12	TABLE ACCESS FULL	INGRESOS	10M	352M		8266 (2)	00:01:40
13	TABLE ACCESS FULL	TICKETS	10M	143M		8478 (2)	00:01:42
14	TABLE ACCESS FULL	LINEASTICKET	29M	572M		35728 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CL"."ID_CLIENTE"=:B1)
   4 - access("P"."ID_PRODUCTO"=:B1)
   5 - filter("LI"."ID_INGRESO"=:B1 AND "LI"."ID_IMPUESTO"=:B2)
   7 - access("PER"."ID_PERSONA"=:B1)
   8 - access("T"."ID_TICKET"="L"."ID_TICKET")
   9 - access("T"."ID_TICKET"="I"."ID_INGRESO")
  10 - access("PG"."ID_INGRESO"="I"."ID_INGRESO")
  11 - filter("PG"."TOTAL"<=800 AND "PG"."TOTAL">=500)
```

ANEXO 136

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 4 SELECT USANDO SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
Plan hash value: 2847663425
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		337K	24M		157K (1)	00:31:28
1	TABLE ACCESS BY INDEX ROWID	CLIENTES	1	33		3 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	CLIENTES_ID_CLIENTE	1			2 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
5	TABLE ACCESS BY INDEX ROWID	PERSONA	1	31		1 (0)	00:00:01
* 6	INDEX UNIQUE SCAN	PERSONA_ID_PERSONA	1			0 (0)	00:00:01
* 7	HASH JOIN		337K	24M	7472K	157K (1)	00:31:28
* 8	HASH JOIN		112K	6150K	5824K	75583 (1)	00:15:08
* 9	HASH JOIN		112K	4502K	4400K	53967 (1)	00:10:48
* 10	HASH JOIN		112K	3075K	2528K	31678 (2)	00:06:21
* 11	TABLE ACCESS FULL	PAGOS	112K	1208K		9486 (2)	00:01:54
12	TABLE ACCESS FULL	INGRESOS	10M	352M		8266 (2)	00:01:40
13	TABLE ACCESS FULL	LINEASIMPUESTO	10M	123M		10165 (2)	00:02:02
14	TABLE ACCESS FULL	TICKETS	10M	143M		8478 (2)	00:01:42
15	TABLE ACCESS FULL	LINEASTICKET	29M	572M		35728 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CL"."ID_CLIENTE"=:B1)
   4 - access("P"."ID_PRODUCTO"=:B1)
   6 - access("PER"."ID_PERSONA"=:B1)
   7 - access("L"."ID_IMPUESTO"="LI"."ID_IMPUESTO" AND "L"."ID_TICKET"="T"."ID_TICKET")
   8 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   9 - access("LI"."ID_INGRESO"="I"."ID_INGRESO")
  10 - access("PG"."ID_INGRESO"="I"."ID_INGRESO")
  11 - filter("PG"."TOTAL"<=800 AND "PG"."TOTAL">=500)
```

ANEXO 137

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 4 SELECT CON ÍNDICE + SUBQUERY 10 MILLONES DE DATOS.

```
Plan hash value: 379305613
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		337K	20M		188K (1)	00:37:45
1	TABLE ACCESS BY INDEX ROWID	CLIENTES	1	33		3 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	CLIENTES_ID_CLIENTE	1			2 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
* 5	TABLE ACCESS FULL	LINEASIMPUESTO	1	13		10169 (2)	00:02:03
6	TABLE ACCESS BY INDEX ROWID	PERSONA	1	31		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	PERSONA_ID_PERSONA	1			0 (0)	00:00:01
* 8	HASH JOIN		337K	20M	6048K	188K (1)	00:37:45
* 9	HASH JOIN		112K	4722K	4400K	107K (1)	00:21:25
* 10	HASH JOIN		112K	3075K	2528K	85453 (1)	00:17:06
11	TABLE ACCESS BY INDEX ROWID	PAGOS	112K	1208K		63261 (1)	00:12:40
* 12	INDEX RANGE SCAN	PAGOS_TOTAL	112K			243 (1)	00:00:03
13	TABLE ACCESS FULL	INGRESOS	10M	352M		8266 (2)	00:01:40
14	TABLE ACCESS FULL	TICKETS	10M	143M		8478 (2)	00:01:42
15	TABLE ACCESS FULL	LINEASTICKET	29M	572M		35728 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CL"."ID_CLIENTE"=:B1)
   4 - access("P"."ID_PRODUCTO"=:B1)
   5 - filter("LI"."ID_INGRESO"=:B1 AND "LI"."ID_IMPUESTO"=:B2)
   7 - access("PER"."ID_PERSONA"=:B1)
   8 - access("T"."ID_TICKET"="L"."ID_TICKET")
   9 - access("T"."ID_TICKET"="I"."ID_INGRESO")
  10 - access("PG"."ID_INGRESO"="I"."ID_INGRESO")
  12 - access("PG"."TOTAL">=500 AND "PG"."TOTAL"<=800)
```

ANEXO 138

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 4 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 10 MILLONES DE DATOS.

```
Plan hash value: 2804176977
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		337K	24M		211K (1)	00:42:14
1	TABLE ACCESS BY INDEX ROWID	CLIENTES	1	33		3 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	CLIENTES_ID_CLIENTE	1			2 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
5	TABLE ACCESS BY INDEX ROWID	PERSONA	1	31		1 (0)	00:00:01
* 6	INDEX UNIQUE SCAN	PERSONA_ID_PERSONA	1			0 (0)	00:00:01
* 7	HASH JOIN		337K	24M	7472K	211K (1)	00:42:14
* 8	HASH JOIN		112K	6150K	5824K	129K (1)	00:25:53
* 9	HASH JOIN		112K	4502K	4400K	107K (1)	00:21:33
* 10	HASH JOIN		112K	3075K	2528K	85453 (1)	00:17:06
11	TABLE ACCESS BY INDEX ROWID	PAGOS	112K	1208K		63261 (1)	00:12:40
* 12	INDEX RANGE SCAN	PAGOS_TOTAL	112K			243 (1)	00:00:03
13	TABLE ACCESS FULL	INGRESOS	10M	352M		8266 (2)	00:01:40
14	TABLE ACCESS FULL	LINEASIMPUESTO	10M	123M		10165 (2)	00:02:02
15	TABLE ACCESS FULL	TICKETS	10M	143M		8478 (2)	00:01:42
16	TABLE ACCESS FULL	LINEASTICKET	29M	572M		35728 (2)	00:07:09

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CL"."ID_CLIENTE"=:B1)
   4 - access("P"."ID_PRODUCTO"=:B1)
   6 - access("PER"."ID_PERSONA"=:B1)
   7 - access("L"."ID_IMPUESTO"="LI"."ID_IMPUESTO" AND "L"."ID_TICKET"="T"."ID_TICKET")
   8 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   9 - access("LI"."ID_INGRESO"="I"."ID_INGRESO")
  10 - access("PG"."ID_INGRESO"="I"."ID_INGRESO")
  12 - access("PG"."TOTAL">=500 AND "PG"."TOTAL"<=800)
```

ANEXO 139

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 1 SELECT SIMPLE Y SELECT USANDO JOIN 20 MILLONES DE DATOS.

Plan hash value: 1639797780

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		1698K	238M		249K (1)	00:49:51
* 1	HASH JOIN		1698K	238M		249K (1)	00:49:51
2	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	1106	42028		5 (0)	00:00:01
* 3	HASH JOIN		1698K	176M	3392K	249K (1)	00:49:51
4	TABLE ACCESS FULL	PRODUCTOS	78763	2461K		208 (1)	00:00:03
* 5	HASH JOIN		1722K	126M	37M	241K (1)	00:48:20
* 6	HASH JOIN		574K	30M	31M	74110 (2)	00:14:50
* 7	HASH JOIN		573K	25M	47M	49039 (2)	00:09:49
* 8	TABLE ACCESS FULL	INGRESOS	1721K	27M		16618 (2)	00:03:20
* 9	HASH JOIN		6666K	184M		17065 (2)	00:03:25
10	NESTED LOOPS		3	60		3 (0)	00:00:01
11	TABLE ACCESS BY INDEX ROWID	ROLES	1	14		1 (0)	00:00:01
* 12	INDEX UNIQUE SCAN	ROLES_NOMBRE	1			0 (0)	00:00:01
* 13	TABLE ACCESS FULL	PERSONA	3	18		2 (0)	00:00:01
14	TABLE ACCESS FULL	TICKETS	20M	839M		16971 (2)	00:03:24
* 15	TABLE ACCESS FULL	LINEASIMPUESTO	2904K	27M		20447 (2)	00:04:06
16	TABLE ACCESS FULL	LINEASTICKET	59M	1201M		71496 (2)	00:14:18

Predicate Information (identified by operation id):

```
   1 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"="L"."ID_ATRIBUTOCONJUNTOINSTANCIA")
   3 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
   5 - access("T"."ID_TICKET"="L"."ID_TICKET")
   6 - access("IM"."ID_INGRESO"="I"."ID_INGRESO")
   7 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   8 - filter("I"."FECHA"<=TIMESTAMP' 2013-01-31 00:00:00' AND "I"."FECHA">=TIMESTAMP' 2013-01-01 00:00:00')
   9 - access("T"."ID_PERSONA"="PS"."ID_PERSONA")
  12 - access("R"."NOMBRE"='Dependiente')
  13 - filter("PS"."ID_ROL"="R"."ID_ROL")
  15 - filter("IM"."MONTO"<500)
```

ANEXO 140

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 1 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 20 MILLONES DE DATOS.

```
Plan hash value: 3434376623
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		1698K	238M		1333K (1)	04:26:45
*1	HASH JOIN		1698K	238M		1333K (1)	04:26:45
2	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	1106	42028		5 (0)	00:00:01
*3	HASH JOIN		1698K	176M	3392K	1333K (1)	04:26:45
4	TABLE ACCESS FULL	PRODUCTOS	78763	2461K		208 (1)	00:00:03
*5	HASH JOIN		1722K	126M	37M	1326K (1)	04:25:14
*6	HASH JOIN		574K	30M	31M	1158K (1)	03:51:44
*7	HASH JOIN		573K	25M	47M	1133K (1)	03:46:43
8	TABLE ACCESS BY INDEX ROWID	INGRESOS	1721K	27M		1101K (1)	03:40:14
*9	INDEX RANGE SCAN	INGRESOS_FECHA	1721K			4574 (1)	00:00:55
*10	HASH JOIN		6666K	184M		17065 (2)	00:03:25
11	NESTED LOOPS		3	60		3 (0)	00:00:01
12	TABLE ACCESS BY INDEX ROWID	ROLES	1	14		1 (0)	00:00:01
*13	INDEX UNIQUE SCAN	ROLES_NOMBRE	1			0 (0)	00:00:01
*14	TABLE ACCESS FULL	PERSONA	3	18		2 (0)	00:00:01
15	TABLE ACCESS FULL	TICKETS	20M	1449M		16971 (2)	00:03:24
*16	TABLE ACCESS FULL	LINEASIMPUESTO	2904K	27M		20447 (2)	00:04:06
17	TABLE ACCESS FULL	LINEASTICKET	59M	1201M		71496 (2)	00:14:18

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"="L"."ID_ATRIBUTOCONJUNTOINSTANCIA")
   3 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
   5 - access("T"."ID_TICKET"="L"."ID_TICKET")
   6 - access("IM"."ID_INGRESO"="I"."ID_INGRESO")
   7 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   9 - access("I"."FECHA">=TIMESTAMP' 2013-01-01 00:00:00' AND "I"."FECHA"<=TIMESTAMP' 2013-01-31 00:00:00')
  10 - access("T"."ID_PERSONA"="PS"."ID_PERSONA")
  13 - access("R"."NOMBRE"='Dependiente')
  14 - filter("PS"."ID_ROL"="R"."ID_ROL")
  16 - filter("IM"."MONTO"<500)
```

ANEXO 141

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 1 SELECT USANDO SUBQUERY 20 MILLONES DE DATOS..

```
Plan hash value: 3849619915
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		5188K	168M		253M (1)	845:23:02
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	ATRIBUTOCONJUNTOINSTANCIA	1	38		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	ATRIBUTOCONJUNTOINSTANCIA_ID_A	1			1 (0)	00:00:01
* 5	FILTER						
* 6	HASH JOIN RIGHT SEMI		5188K	168M	41M	210K (1)	00:42:03
7	VIEW	VW_NSO_1	1722K	21M		42485 (2)	00:08:30
* 8	HASH JOIN		1722K	44M	47M	42485 (2)	00:08:30
* 9	TABLE ACCESS FULL	INGRESOS	1721K	27M		16618 (2)	00:03:20
* 10	TABLE ACCESS FULL	LINEASIMPUESTO	2904K	27M		20447 (2)	00:04:06
11	TABLE ACCESS FULL	LINEASTICKET	59M	1201M		71550 (2)	00:14:19
12	NESTED LOOPS						
13	NESTED LOOPS		2	110		49 (0)	00:00:01
14	NESTED LOOPS		21	966		7 (0)	00:00:01
15	NESTED LOOPS		1	29		5 (0)	00:00:01
16	NESTED LOOPS		1	23		4 (0)	00:00:01
17	TABLE ACCESS BY INDEX ROWID	TICKETS	1	9		3 (0)	00:00:01
* 18	INDEX UNIQUE SCAN	TICKETS_ID_TICKET	1			2 (0)	00:00:01
19	TABLE ACCESS BY INDEX ROWID	ROLES	1	14		1 (0)	00:00:01
* 20	INDEX UNIQUE SCAN	ROLES_NOMBRE	1			0 (0)	00:00:01
* 21	TABLE ACCESS BY INDEX ROWID	PERSONA	10	60		1 (0)	00:00:01
* 22	INDEX UNIQUE SCAN	PERSONA_ID_PERSONA	1			0 (0)	00:00:01
* 23	TABLE ACCESS FULL	INGRESOS	21	357		2 (0)	00:00:01
* 24	INDEX UNIQUE SCAN	TICKETS_ID_TICKET	1			1 (0)	00:00:01
* 25	TABLE ACCESS BY INDEX ROWID	TICKETS	1	9		2 (0)	00:00:01

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   4 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"=:B1)
   5 - filter( EXISTS (SELECT 0 FROM "TICKETS" "TICKETS","PERSONA" "PERSONA","TICKETS" "TICKETS","INGRESOS"
              "INGRESOS","ROLES" "ROLES" WHERE "NOMBRE"='Dependiente' AND "FECHA"<=TIMESTAMP' 2013-01-31 00:00:00' AND
              "FECHA">=TIMESTAMP' 2013-01-01 00:00:00' AND "ID_TICKET"="ID_INGRESO" AND "ID_PERSONA"="ID_PERSONA" AND
              "ID_PERSONA"="ID_PERSONA" AND "ID_ROL"="ID_ROL" AND "ID_TICKET"=:B1))
   6 - access("ID_TICKET"="ID_INGRESO")
   8 - access("ID_INGRESO"="ID_INGRESO")
   9 - filter("FECHA"<=TIMESTAMP' 2013-01-31 00:00:00' AND "FECHA">=TIMESTAMP' 2013-01-01 00:00:00')
  10 - filter("MONTO"<500)
  18 - access("ID_TICKET"=:B1)
  20 - access("NOMBRE"='Dependiente')
  21 - filter("ID_ROL"="ID_ROL")
  22 - access("ID_PERSONA"="ID_PERSONA")
  23 - filter("FECHA"<=TIMESTAMP' 2013-01-31 00:00:00' AND "FECHA">=TIMESTAMP' 2013-01-01 00:00:00')
  24 - access("ID_TICKET"="ID_INGRESO")
  25 - filter("ID_PERSONA"="ID_PERSONA")
```

ANEXO 142

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 1 SELECT USANDO SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
Plan hash value: 2070554531
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		1722K	126M		241K (1)	00:48:20
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	ATRIBUTOCONJUNTOINSTANCIA	1	38		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	ATRIBUTOCONJUNTOINSTANCIA_ID_A	1			1 (0)	00:00:01
* 5	HASH JOIN		1722K	126M	37M	241K (1)	00:48:20
* 6	HASH JOIN		574K	30M	31M	74110 (2)	00:14:50
* 7	HASH JOIN		573K	25M	47M	49039 (2)	00:09:49
* 8	TABLE ACCESS FULL	INGRESOS	1721K	27M		16618 (2)	00:03:20
* 9	HASH JOIN		6666K	184M		17065 (2)	00:03:25
10	NESTED LOOPS		3	60		3 (0)	00:00:01
11	TABLE ACCESS BY INDEX ROWID	ROLES	1	14		1 (0)	00:00:01
* 12	INDEX UNIQUE SCAN	ROLES_NOMBRE	1			0 (0)	00:00:01
* 13	TABLE ACCESS FULL	PERSONA	3	18		2 (0)	00:00:01
14	TABLE ACCESS FULL	TICKETS	20M	171M		16971 (2)	00:03:24
* 15	TABLE ACCESS FULL	LINEASIMPUESTO	2904K	27M		20447 (2)	00:04:06
16	TABLE ACCESS FULL	LINEASTICKET	59M	1201M		71496 (2)	00:14:18

```
Predicate Information (identified by operation id):
---------------------------------------------------

2 - access("P"."ID_PRODUCTO"=:B1)
4 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"=:B1)
5 - access("T"."ID_TICKET"="L"."ID_TICKET")
6 - access("IM"."ID_INGRESO"="I"."ID_INGRESO")
7 - access("T"."ID_TICKET"="I"."ID_INGRESO")
8 - filter("I"."FECHA"<=TIMESTAMP' 2013-01-31 00:00:00' AND "I"."FECHA">=TIMESTAMP' 2013-01-01 00:00:00')
9 - access("PS"."ID_PERSONA"="T"."ID_PERSONA")
12 - access("R"."NOMBRE"='Dependiente')
13 - filter("PS"."ID_ROL"="R"."ID_ROL")
15 - filter("IM"."MONTO"<500)
```

ANEXO 143

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 1 SELECT USANDO ÍNDICE + SUBQUERY 20 MILLONES DE DATOS.

Plan hash value: 1511226799

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		5188K	168M		327M (1)	999:59:59
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	ATRIBUTOCONJUNTOINSTANCIA	1	38		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	ATRIBUTOCONJUNTOINSTANCIA_ID_A	1			1 (0)	00:00:01
* 5	FILTER						
* 6	HASH JOIN RIGHT SEMI		5188K	168M	41M	1294K (1)	04:18:57
7	VIEW	VW_NSO_1	1722K	21M		1126K (1)	03:45:24
* 8	HASH JOIN		1722K	44M	47M	1126K (1)	03:45:24
9	TABLE ACCESS BY INDEX ROWID	INGRESOS	1721K	27M		1101K (1)	03:40:14
* 10	INDEX RANGE SCAN	INGRESOS_FECHA	1721K			4574 (1)	00:00:55
* 11	TABLE ACCESS FULL	LINEASIMPUESTO	2904K	27M		20447 (2)	00:04:06
12	TABLE ACCESS FULL	LINEASTICKET	59M	1201M		71550 (2)	00:14:19
13	NESTED LOOPS						
14	NESTED LOOPS		2	110		63 (0)	00:00:01
15	NESTED LOOPS		21	966		21 (0)	00:00:01
16	NESTED LOOPS		1	29		5 (0)	00:00:01
17	NESTED LOOPS		1	23		4 (0)	00:00:01
18	TABLE ACCESS BY INDEX ROWID	TICKETS	1	9		3 (0)	00:00:01
* 19	INDEX UNIQUE SCAN	TICKETS_ID_TICKET	1			2 (0)	00:00:01
20	TABLE ACCESS BY INDEX ROWID	ROLES	1	14		1 (0)	00:00:01
* 21	INDEX UNIQUE SCAN	ROLES_NOMBRE	1			0 (0)	00:00:01
* 22	TABLE ACCESS BY INDEX ROWID	PERSONA	10	60		1 (0)	00:00:01
* 23	INDEX UNIQUE SCAN	PERSONA_ID_PERSONA	1			0 (0)	00:00:01
24	TABLE ACCESS BY INDEX ROWID	INGRESOS	21	357		16 (0)	00:00:01
* 25	INDEX RANGE SCAN	INGRESOS_FECHA	21			2 (0)	00:00:01
* 26	INDEX UNIQUE SCAN	TICKETS_ID_TICKET	1			1 (0)	00:00:01
* 27	TABLE ACCESS BY INDEX ROWID	TICKETS	1	9		2 (0)	00:00:01

```
Predicate Information (identified by operation id):
---------------------------------------------------
    2 - access("P"."ID_PRODUCTO"=:B1)
   4 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"=:B1)
   5 - filter( EXISTS (SELECT /*+ INDEX ("INGRESOS" "INGRESOS_FECHA") */ 0 FROM "TICKETS" "TICKETS","PERSONA"
              "PERSONA","TICKETS" "TICKETS","INGRESOS" "INGRESOS","ROLES" "ROLES" WHERE "NOMBRE"='Dependiente' AND
              "FECHA"<=TIMESTAMP' 2013-01-31 00:00:00' AND "FECHA">=TIMESTAMP' 2013-01-01 00:00:00' AND "ID_TICKET"="ID_INGRESO"
              AND "ID_PERSONA"="ID_PERSONA" AND "ID_PERSONA"="ID_PERSONA" AND "ID_ROL"="ID_ROL" AND "ID_TICKET"=:B1))
   6 - access("ID_TICKET"="ID_INGRESO")
   8 - access("ID_INGRESO"="ID_INGRESO")
  10 - access("FECHA">=TIMESTAMP' 2013-01-01 00:00:00' AND "FECHA"<=TIMESTAMP' 2013-01-31 00:00:00')
  11 - filter("MONTO"<500)
  19 - access("ID_TICKET"=:B1)
  21 - access("NOMBRE"='Dependiente')
  22 - filter("ID_ROL"="ID_ROL")
  23 - access("ID_PERSONA"="ID_PERSONA")
  25 - access("FECHA">=TIMESTAMP' 2013-01-01 00:00:00' AND "FECHA"<=TIMESTAMP' 2013-01-31 00:00:00')
  26 - access("ID_TICKET"="ID_INGRESO")
  27 - filter("ID_PERSONA"="ID_PERSONA")
```

ANEXO 144

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 1 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
Plan hash value: 2538894714
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		1722K	126M		1326K (1)	04:25:14
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	ATRIBUTOCONJUNTOINSTANCIA	1	38		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	ATRIBUTOCONJUNTOINSTANCIA_ID_A	1			1 (0)	00:00:01
* 5	HASH JOIN		1722K	126M	37M	1326K (1)	04:25:14
* 6	HASH JOIN		574K	30M	31M	1158K (1)	03:51:44
* 7	HASH JOIN		573K	25M	47M	1133K (1)	03:46:43
8	TABLE ACCESS BY INDEX ROWID	INGRESOS	1721K	27M		1101K (1)	03:40:14
* 9	INDEX RANGE SCAN	INGRESOS_FECHA	1721K			4574 (1)	00:00:55
* 10	HASH JOIN		6666K	184M		17065 (2)	00:03:25
11	NESTED LOOPS		3	60		3 (0)	00:00:01
12	TABLE ACCESS BY INDEX ROWID	ROLES	1	14		1 (0)	00:00:01
* 13	INDEX UNIQUE SCAN	ROLES_NOMBRE	1			0 (0)	00:00:01
* 14	TABLE ACCESS FULL	PERSONA	3	18		2 (0)	00:00:01
15	TABLE ACCESS FULL	TICKETS	20M	171M		16971 (2)	00:03:24
* 16	TABLE ACCESS FULL	LINEASIMPUESTO	2904K	27M		20447 (2)	00:04:06
17	TABLE ACCESS FULL	LINEASTICKET	59M	1201M		71496 (2)	00:14:18

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   4 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"=:B1)
   5 - access("T"."ID_TICKET"="L"."ID_TICKET")
   6 - access("IM"."ID_INGRESO"="I"."ID_INGRESO")
   7 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   9 - access("I"."FECHA">=TIMESTAMP' 2013-01-01 00:00:00' AND "I"."FECHA"<=TIMESTAMP' 2013-01-31 00:00:00')
  10 - access("PS"."ID_PERSONA"="T"."ID_PERSONA")
  13 - access("R"."NOMBRE"='Dependiente')
  14 - filter("PS"."ID_ROL"="R"."ID_ROL")
  16 - filter("IM"."MONTO"<500)
```

ANEXO 145

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 2 SELECT SIMPLE Y SELECT USANDO JOIN 20 MILLONES DE DATOS.

```
Plan hash value: 2879677815
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		134K	21M		131K (2)	00:26:17
*1	HASH JOIN		134K	21M	17M	131K (2)	00:26:17
*2	HASH JOIN		135K	15M	15M	113K (2)	00:22:46
*3	HASH JOIN		135K	13M	3848K	73180 (2)	00:14:39
4	TABLE ACCESS FULL	PRODUCTOS	78763	2922K		208 (1)	00:00:03
*5	HASH JOIN		273K	18M		71719 (2)	00:14:21
6	NESTED LOOPS		5	260		6 (0)	00:00:01
7	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	1	14		1 (0)	00:00:01
*8	INDEX UNIQUE SCAN	CATEGORIAS_NOMBRE	1			0 (0)	00:00:01
*9	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	5	190		5 (0)	00:00:01
10	TABLE ACCESS FULL	LINEASTICKET	59M	1029M		71442 (1)	00:14:18
11	TABLE ACCESS FULL	TICKETS	20M	228M		16990 (2)	00:03:24
12	TABLE ACCESS FULL	CLIENTES	2240K	102M		10343 (1)	00:02:05

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("CL"."ID_CLIENTE"="T"."ID_CLIENTE")
   2 - access("T"."ID_TICKET"="L"."ID_TICKET")
   3 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO" AND "C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   5 - access("ACI"."ID_ATRIBUTOCONJUNTOINSTANCIA"="L"."ID_ATRIBUTOCONJUNTOINSTANCIA")
   8 - access("C"."NOMBRE"='FARMACEUTICOS')
   9 - filter("ACI"."DESCRIPCION"='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR
              "ACI"."DESCRIPCION"='ATRIB.CONJUNTO NATURPHARMA' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX GENERICOS'
              OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO VITAFARMA')
```

ANEXO 146

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 2 SELECT USANDO ÍNDICE 20 MILLONES DE DATOS.

```
Plan hash value: 1758223253
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		134K	21M		305K (1)	01:01:04
* 1	HASH JOIN		134K	21M	17M	305K (1)	01:01:04
* 2	HASH JOIN		135K	15M	15M	287K (1)	00:57:34
* 3	HASH JOIN		135K	13M	3848K	247K (1)	00:49:26
4	TABLE ACCESS FULL	PRODUCTOS	78763	2922K		208 (1)	00:00:03
5	NESTED LOOPS						
6	NESTED LOOPS		273K	18M		245K (1)	00:49:08
7	NESTED LOOPS		5	260		6 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	1	14		1 (0)	00:00:01
* 9	INDEX UNIQUE SCAN	CATEGORIAS_NOMBRE	1			0 (0)	00:00:01
* 10	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	5	190		5 (0)	00:00:01
* 11	INDEX RANGE SCAN	ATRIBCONJINST_DESCRIP	54246			114 (1)	00:00:02
12	TABLE ACCESS BY INDEX ROWID	LINEASTICKET	54246	953K		49132 (1)	00:09:50
13	TABLE ACCESS FULL	TICKETS	20M	228M		16990 (2)	00:03:24
14	TABLE ACCESS FULL	CLIENTES	2240K	102M		10343 (1)	00:02:05

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("CL"."ID_CLIENTE"="T"."ID_CLIENTE")
   2 - access("T"."ID_TICKET"="L"."ID_TICKET")
   3 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO" AND "C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   9 - access("C"."NOMBRE"='FARMACEUTICOS')
  10 - filter("ACI"."DESCRIPCION"='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO
              NATURPHARMA' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO
              QUIFATEX PROP MERZ' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO VITAFARMA')
  11 - access("ACI"."ID_ATRIBUTOCONJUNTOINSTANCIA"="L"."ID_ATRIBUTOCONJUNTOINSTANCIA")
```

ANEXO 147

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 2 SELECT USANDO SUBQUERY 20 MILLONES DE DATOS..

```
Plan hash value: 3539947530
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		272K	30M		130K (2)	00:26:05
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	NESTED LOOPS		1	22		3 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 9	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
10	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 11	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
12	TABLE ACCESS BY INDEX ROWID	ATRIBUTOCONJUNTOINSTANCIA	1	38		2 (0)	00:00:01
* 13	INDEX UNIQUE SCAN	ATRIBUTOCONJUNTOINSTANCIA_ID_A	1			1 (0)	00:00:01
* 14	FILTER						
* 15	HASH JOIN		272K	30M	20M	130K (2)	00:26:05
* 16	HASH JOIN		273K	17M	17M	112K (2)	00:22:32
* 17	HASH JOIN		273K	14M		71854 (2)	00:14:23
* 18	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	5	190		5 (0)	00:00:01
19	TABLE ACCESS FULL	LINEASTICKET	59M	1029M		71577 (2)	00:14:19
20	TABLE ACCESS FULL	TICKETS	20M	228M		16990 (2)	00:03:24
21	TABLE ACCESS FULL	CLIENTES	2240K	102M		10343 (1)	00:02:05

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   5 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   9 - access("P"."ID_PRODUCTO"=:B1)
  11 - access("P"."ID_PRODUCTO"=:B1)
  13 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"=:B1)
  14 - filter( (SELECT "C"."NOMBRE" FROM "CATEGORIAS" "C","PRODUCTOS" "P" WHERE "P"."ID_PRODUCTO"=:B1 AND
              "C"."ID_CATEGORIA"="P"."ID_CATEGORIA")='FARMACEUTICOS')
  15 - access("T"."ID_CLIENTE"="CL"."ID_CLIENTE")
  16 - access("L"."ID_TICKET"="T"."ID_TICKET")
  17 - access("L"."ID_ATRIBUTOCONJUNTOINSTANCIA"="A"."ID_ATRIBUTOCONJUNTOINSTANCIA")
  18 - filter("DESCRIPCION"='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR "DESCRIPCION"='ATRIB.CONJUNTO
              NATURPHARMA' OR "DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR "DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX
              PROP MERZ' OR "DESCRIPCION"='ATRIB.CONJUNTO VITAFARMA')
```

ANEXO 148

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 2 SELECT USANDO SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
Plan hash value: 2879677815
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		134K	21M		131K (2)	00:26:17
* 1	HASH JOIN		134K	21M	17M	131K (2)	00:26:17
* 2	HASH JOIN		135K	15M	15M	113K (2)	00:22:46
* 3	HASH JOIN		135K	13M	3848K	73180 (2)	00:14:39
4	TABLE ACCESS FULL	PRODUCTOS	78763	2922K		208 (1)	00:00:03
* 5	HASH JOIN		273K	18M		71719 (2)	00:14:21
6	NESTED LOOPS		5	260		6 (0)	00:00:01
7	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	1	14		1 (0)	00:00:01
* 8	INDEX UNIQUE SCAN	CATEGORIAS_NOMBRE	1			0 (0)	00:00:01
* 9	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	5	190		5 (0)	00:00:01
10	TABLE ACCESS FULL	LINEASTICKET	59M	1029M		71442 (1)	00:14:18
11	TABLE ACCESS FULL	TICKETS	20M	228M		16990 (2)	00:03:24
12	TABLE ACCESS FULL	CLIENTES	2240K	102M		10343 (1)	00:02:05

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("CL"."ID_CLIENTE"="T"."ID_CLIENTE")
   2 - access("L"."ID_TICKET"="T"."ID_TICKET")
   3 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA" AND "P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
   5 - access("ACI"."ID_ATRIBUTOCONJUNTOINSTANCIA"="L"."ID_ATRIBUTOCONJUNTOINSTANCIA")
   8 - access("C"."NOMBRE"='FARMACEUTICOS')
   9 - filter("ACI"."DESCRIPCION"='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR
              "ACI"."DESCRIPCION"='ATRIB.CONJUNTO NATURPHARMA' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX GENERICOS'
              OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO VITAFARMA')
```

ANEXO 149

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 2 SELECT USANDO ÍNDICE + SUBQUERY 20 MILLONES DE DATOS.

```
Plan hash value: 3539947530
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		272K	30M		130K (2)	00:26:05
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	NESTED LOOPS		1	22		3 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 9	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
10	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 11	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
12	TABLE ACCESS BY INDEX ROWID	ATRIBUTOCONJUNTOINSTANCIA	1	38		2 (0)	00:00:01
* 13	INDEX UNIQUE SCAN	ATRIBUTOCONJUNTOINSTANCIA_ID_A	1			1 (0)	00:00:01
* 14	FILTER						
* 15	HASH JOIN		272K	30M	20M	130K (2)	00:26:05
* 16	HASH JOIN		273K	17M	17M	112K (2)	00:22:32
* 17	HASH JOIN		273K	14M		71854 (2)	00:14:23
* 18	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	5	190		5 (0)	00:00:01
19	TABLE ACCESS FULL	LINEASTICKET	59M	1029M		71577 (2)	00:14:19
20	TABLE ACCESS FULL	TICKETS	20M	228M		16990 (2)	00:03:24
21	TABLE ACCESS FULL	CLIENTES	2240K	102M		10343 (1)	00:02:05

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   5 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   9 - access("P"."ID_PRODUCTO"=:B1)
  11 - access("P"."ID_PRODUCTO"=:B1)
  13 - access("A"."ID_ATRIBUTOCONJUNTOINSTANCIA"=:B1)
  14 - filter( (SELECT "C"."NOMBRE" FROM "CATEGORIAS" "C","PRODUCTOS" "P" WHERE "P"."ID_PRODUCTO"=:B1 AND
              "C"."ID_CATEGORIA"="P"."ID_CATEGORIA")='FARMACEUTICOS')
  15 - access("T"."ID_CLIENTE"="CL"."ID_CLIENTE")
  16 - access("L"."ID_TICKET"="T"."ID_TICKET")
  17 - access("L"."ID_ATRIBUTOCONJUNTOINSTANCIA"="A"."ID_ATRIBUTOCONJUNTOINSTANCIA")
  18 - filter("DESCRIPCION"='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR "DESCRIPCION"='ATRIB.CONJUNTO
             NATURPHARMA' OR "DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX GENERICOS' OR "DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX
             PROP MERZ' OR "DESCRIPCION"='ATRIB.CONJUNTO VITAFARMA')
```

ANEXO 150

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 2 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
Plan hash value: 821077450
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		272K	90M		304K (1)	01:00:51
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	NESTED LOOPS		1	22		3 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 9	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
10	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 11	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
12	VIEW		272K	90M		304K (1)	01:00:51
* 13	FILTER						
* 14	HASH JOIN		272K	30M	20M	304K (1)	01:00:51
* 15	HASH JOIN		273K	17M	17M	286K (1)	00:57:17
16	NESTED LOOPS						
17	NESTED LOOPS		273K	14M		245K (1)	00:49:08
* 18	TABLE ACCESS FULL	ATRIBUTOCONJUNTOINSTANCIA	5	190		5 (0)	00:00:01
* 19	INDEX RANGE SCAN	ATRIBCONJINST_DESCRIP	54246			114 (1)	00:00:02
20	TABLE ACCESS BY INDEX ROWID	LINEASTICKET	54246	953K		49132 (1)	00:09:50
21	TABLE ACCESS FULL	TICKETS	20M	228M		16990 (2)	00:03:24
22	TABLE ACCESS FULL	CLIENTES	2240K	102M		10343 (1)	00:02:05

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   5 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   9 - access("P"."ID_PRODUCTO"=:B1)
  11 - access("P"."ID_PRODUCTO"=:B1)
  13 - filter( (SELECT "C"."NOMBRE" FROM "CATEGORIAS" "C","PRODUCTOS" "P" WHERE "P"."ID_PRODUCTO"=:B1 AND
              "C"."ID_CATEGORIA"="P"."ID_CATEGORIA")='FARMACEUTICOS')
  14 - access("T"."ID_CLIENTE"="CL"."ID_CLIENTE")
  15 - access("L"."ID_TICKET"="T"."ID_TICKET")
  18 - filter("ACI"."DESCRIPCION"='ATRIB.CONJUNTO GRUNENTHAL-GENERICOS' OR
              "ACI"."DESCRIPCION"='ATRIB.CONJUNTO NATURPHARMA' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX GENERICOS'
              OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO QUIFATEX PROP MERZ' OR "ACI"."DESCRIPCION"='ATRIB.CONJUNTO VITAFARMA')
  19 - access("ACI"."ID_ATRIBUTOCONJUNTOINSTANCIA"="L"."ID_ATRIBUTOCONJUNTOINSTANCIA")
```

ANEXO 151

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 3 SELECT SIMPLE 20 MILLONES DE DATOS.

```
Plan hash value: 4030141452
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		50887	4422K		420K (1)	01:24:01
* 1	FILTER						
2	HASH GROUP BY		50887	4422K	1968M	420K (1)	01:24:01
* 3	HASH JOIN		20M	1727M		272K (1)	00:54:30
4	TABLE ACCESS FULL	CATEGORIAS	2	28		2 (0)	00:00:01
* 5	HASH JOIN		20M	1455M	3616K	272K (1)	00:54:29
6	TABLE ACCESS FULL	PRODUCTOS	78763	2692K		208 (1)	00:00:03
* 7	HASH JOIN		20M	787M	229M	221K (1)	00:44:13
* 8	HASH JOIN		6878K	150M	190M	55355 (2)	00:11:05
* 9	TABLE ACCESS FULL	INGRESOS	6878K	111M		16644 (2)	00:03:20
10	INDEX FAST FULL SCAN	TICKETS_ID_TICKET	20M	114M		12033 (2)	00:02:25
11	TABLE ACCESS FULL	LINEASTICKET	59M	972M		71496 (2)	00:14:18

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(SUM("L"."UNIDADES")>=600 AND SUM("L"."UNIDADES")<=1000)
   3 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   5 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
   7 - access("T"."ID_TICKET"="L"."ID_TICKET")
   8 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   9 - filter("I"."FECHA">=TIMESTAMP' 2013-05-01 00:00:00' AND "I"."FECHA"<=TIMESTAMP'
              2013-08-31 00:00:00')
```

ANEXO 152

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 3 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 20 MILLONES DE DATOS.

```
Plan hash value: 3365292073
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		50887	4422K		4802K (1)	16:00:32
* 1	FILTER						
2	HASH GROUP BY		50887	4422K	1968M	4802K (1)	16:00:32
* 3	HASH JOIN		20M	1727M		4654K (1)	15:31:00
4	TABLE ACCESS FULL	CATEGORIAS	2	28		2 (0)	00:00:01
* 5	HASH JOIN		20M	1455M	3616K	4654K (1)	15:30:59
6	TABLE ACCESS FULL	PRODUCTOS	78763	2692K		208 (1)	00:00:03
* 7	HASH JOIN		20M	787M	229M	4603K (1)	15:20:44
* 8	HASH JOIN		6878K	150M	190M	4437K (1)	14:47:35
9	TABLE ACCESS BY INDEX ROWID	INGRESOS	6878K	111M		4399K (1)	14:39:50
* 10	INDEX RANGE SCAN	INGRESOS_FECHA	6878K			18267 (1)	00:03:40
11	INDEX FAST FULL SCAN	TICKETS_ID_TICKET	20M	114M		12033 (2)	00:02:25
12	TABLE ACCESS FULL	LINEASTICKET	59M	972M		71496 (2)	00:14:18

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(SUM("L"."UNIDADES")>=600 AND SUM("L"."UNIDADES")<=1000)
   3 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   5 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
   7 - access("T"."ID_TICKET"="L"."ID_TICKET")
   8 - access("T"."ID_TICKET"="I"."ID_INGRESO")
  10 - access("I"."FECHA">=TIMESTAMP' 2013-05-01 00:00:00' AND "I"."FECHA"<=TIMESTAMP' 2013-08-31
              00:00:00')
```

ANEXO 153

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 3 SELECT USANDO SUBQUERY 20 MILLONES DE DATOS.

```
Plan hash value: 3356598634
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		47705	1071K		266K (2)	00:53:16
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	NESTED LOOPS		1	22		3 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 9	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
* 10	FILTER						
11	HASH GROUP BY		47705	1071K	636M	266K (2)	00:53:16
* 12	HASH JOIN RIGHT SEMI		20M	454M	118M	215K (1)	00:43:07
13	VIEW	VW_NSO_1	6878K	39M		55355 (2)	00:11:05
* 14	HASH JOIN		6878K	150M	190M	55355 (2)	00:11:05
* 15	TABLE ACCESS FULL	INGRESOS	6878K	111M		16644 (2)	00:03:20
16	INDEX FAST FULL SCAN	TICKETS_ID_TICKET	20M	114M		12033 (2)	00:02:25
17	TABLE ACCESS FULL	LINEASTICKET	59M	972M		71496 (2)	00:14:18

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   5 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   9 - access("P"."ID_PRODUCTO"=:B1)
  10 - filter(SUM("L"."UNIDADES")>=600 AND SUM("L"."UNIDADES")<=1000)
  12 - access("L"."ID_TICKET"="ID_TICKET")
  14 - access("ID_TICKET"="ID_INGRESO")
  15 - filter("FECHA">=TIMESTAMP' 2013-05-01 00:00:00' AND "FECHA"<=TIMESTAMP' 2013-08-31 00:00:00')
```

ANEXO 154

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 3 SELECT USANDO JOIN 20 MILLONES DE DATOS.

```
Plan hash value: 4030141452
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		22M	4150M		459K (1)	01:31:59
* 1	FILTER						
2	HASH GROUP BY		22M	4150M		459K (1)	01:31:59
* 3	HASH JOIN		22M	4150M		458K (1)	01:31:45
4	TABLE ACCESS FULL	CATEGORIAS	2	44		2 (0)	00:00:01
* 5	HASH JOIN		22M	3672M	6712K	458K (1)	01:31:44
6	TABLE ACCESS FULL	PRODUCTOS	76298	5811K		208 (1)	00:00:03
* 7	HASH JOIN		22M	1987M	360M	346K (1)	01:09:16
* 8	HASH JOIN		7407K	275M	268M	66348 (1)	00:13:17
* 9	TABLE ACCESS FULL	INGRESOS	7407K	183M		16640 (2)	00:03:20
10	INDEX FAST FULL SCAN	TICKETS_ID_TICKET	20M	251M		12169 (2)	00:02:27
11	TABLE ACCESS FULL	LINEASTICKET	62M	3113M		71528 (2)	00:14:19

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - filter(SUM("L"."UNIDADES")>=600 AND SUM("L"."UNIDADES")<=1000)
   3 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   5 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
   7 - access("L"."ID_TICKET"="T"."ID_TICKET")
   8 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   9 - filter("I"."FECHA">=TIMESTAMP' 2013-05-01 00:00:00' AND "I"."FECHA"<=TIMESTAMP'
              2013-08-31 00:00:00')
```

ANEXO 155

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 3 SELECT USANDO SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
Plan hash value: 3504110572
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		50887	3577K		395K (1)	01:19:07
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	NESTED LOOPS		1	22		3 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
* 8	FILTER						
9	HASH GROUP BY		50887	3577K	1562M	395K (1)	01:19:07
* 10	HASH JOIN		20M	1397M	3392K	272K (1)	00:54:29
11	TABLE ACCESS FULL	PRODUCTOS	78763	2461K		208 (1)	00:00:03
* 12	HASH JOIN		20M	787M	229M	221K (1)	00:44:13
* 13	HASH JOIN		6878K	150M	190M	55355 (2)	00:11:05
* 14	TABLE ACCESS FULL	INGRESOS	6878K	111M		16644 (2)	00:03:20
15	INDEX FAST FULL SCAN	TICKETS_ID_TICKET	20M	114M		12033 (2)	00:02:25
16	TABLE ACCESS FULL	LINEASTICKET	59M	972M		71496 (2)	00:14:18

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   5 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   8 - filter(SUM("L"."UNIDADES")>=600 AND SUM("L"."UNIDADES")<=1000)
  10 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
  12 - access("L"."ID_TICKET"="T"."ID_TICKET")
  13 - access("T"."ID_TICKET"="I"."ID_INGRESO")
  14 - filter("I"."FECHA">=TIMESTAMP' 2013-05-01 00:00:00' AND "I"."FECHA"<=TIMESTAMP' 2013-08-31
              00:00:00')
```

ANEXO 156

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 3 SELECT USANDO ÍNDICE + SUBQUERY 20 MILLONES DE DATOS.

```
Plan hash value: 87719989
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		47705	1071K		4648K (1)	15:29:46
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
5	NESTED LOOPS		1	22		3 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
8	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 9	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
* 10	FILTER						
11	HASH GROUP BY		47705	1071K	636M	4648K (1)	15:29:46
* 12	HASH JOIN RIGHT SEMI		20M	454M	118M	4598K (1)	15:19:37
13	VIEW	VW_NSO_1	6878K	39M		4437K (1)	14:47:35
* 14	HASH JOIN		6878K	150M	190M	4437K (1)	14:47:35
15	TABLE ACCESS BY INDEX ROWID	INGRESOS	6878K	111M		4399K (1)	14:39:50
* 16	INDEX RANGE SCAN	INGRESOS_FECHA	6878K			18267 (1)	00:03:40
17	INDEX FAST FULL SCAN	TICKETS_ID_TICKET	20M	114M		12033 (2)	00:02:25
18	TABLE ACCESS FULL	LINEASTICKET	59M	972M		71496 (2)	00:14:18

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   4 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("P"."ID_PRODUCTO"=:B1)
   9 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
  10 - filter(SUM("L"."UNIDADES")>=600 AND SUM("L"."UNIDADES")<=1000)
  12 - access("L"."ID_TICKET"="ID_TICKET")
  14 - access("ID_TICKET"="ID_INGRESO")
  16 - access("FECHA">=TIMESTAMP' 2013-05-01 00:00:00' AND "FECHA"<=TIMESTAMP' 2013-08-31 00:00:00')
```

ANEXO 157

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 3 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
Plan hash value: 3889954275
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		50887	3577K		4778K (1)	15:55:37
1	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
3	NESTED LOOPS		1	22		3 (0)	00:00:01
4	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	8		2 (0)	00:00:01
* 5	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
6	TABLE ACCESS BY INDEX ROWID	CATEGORIAS	2	28		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	CATEGORIAS_ID_CATEGORIA	1			0 (0)	00:00:01
* 8	FILTER						
9	HASH GROUP BY		50887	3577K	1562M	4778K (1)	15:55:37
* 10	HASH JOIN		20M	1397M	3392K	4654K (1)	15:30:59
11	TABLE ACCESS FULL	PRODUCTOS	78763	2461K		208 (1)	00:00:03
* 12	HASH JOIN		20M	787M	229M	4603K (1)	15:20:44
* 13	HASH JOIN		6878K	150M	190M	4437K (1)	14:47:35
14	TABLE ACCESS BY INDEX ROWID	INGRESOS	6878K	111M		4399K (1)	14:39:50
* 15	INDEX RANGE SCAN	INGRESOS_FECHA	6878K			18267 (1)	00:03:40
16	INDEX FAST FULL SCAN	TICKETS_ID_TICKET	20M	114M		12033 (2)	00:02:25
17	TABLE ACCESS FULL	LINEASTICKET	59M	972M		71496 (2)	00:14:18

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("P"."ID_PRODUCTO"=:B1)
   5 - access("P"."ID_PRODUCTO"=:B1)
   7 - access("C"."ID_CATEGORIA"="P"."ID_CATEGORIA")
   8 - filter(SUM("L"."UNIDADES")>=600 AND SUM("L"."UNIDADES")<=1000)
  10 - access("P"."ID_PRODUCTO"="L"."ID_PRODUCTO")
  12 - access("L"."ID_TICKET"="T"."ID_TICKET")
  13 - access("T"."ID_TICKET"="I"."ID_INGRESO")
  15 - access("FECHA">=TIMESTAMP' 2013-05-01 00:00:00' AND "FECHA"<=TIMESTAMP' 2013-08-31 00:00:00')
```

ANEXO 158

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 4 SELECT SIMPLE Y SELECT USANDO JOIN 20 MILLONES DE DATOS.

```
Plan hash value: 4211421244
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		560K	91M		334K (1)	01:06:52
* 1	HASH JOIN		560K	91M		334K (1)	01:06:52
2	TABLE ACCESS FULL	PERSONA	10	310		2 (0)	00:00:01
* 3	HASH JOIN		560K	75M	3392K	334K (1)	01:06:52
4	TABLE ACCESS FULL	PRODUCTOS	78763	2461K		208 (1)	00:00:03
* 5	HASH JOIN		568K	59M	18M	330K (1)	01:06:09
* 6	HASH JOIN		189K	16M	12M	166K (1)	00:33:24
* 7	HASH JOIN		190K	10M	9864K	151K (1)	00:30:15
* 8	HASH JOIN		190K	7630K	7448K	108K (1)	00:21:37
* 9	HASH JOIN		190K	5210K	4288K	63424 (2)	00:12:42
* 10	TABLE ACCESS FULL	PAGOS	190K	2047K		19044 (2)	00:03:49
11	TABLE ACCESS FULL	INGRESOS	20M	534M		16569 (2)	00:03:19
12	TABLE ACCESS FULL	LINEASIMPUESTO	20M	247M		20402 (2)	00:04:05
13	TABLE ACCESS FULL	TICKETS	20M	820M		16990 (2)	00:03:24
14	TABLE ACCESS FULL	CLIENTES	2240K	70M		10343 (1)	00:02:05
15	TABLE ACCESS FULL	LINEASTICKET	59M	1144M		71550 (2)	00:14:19

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("T"."ID_PERSONA"="PER"."ID_PERSONA")
   3 - access("L"."ID_PRODUCTO"="P"."ID_PRODUCTO")
   5 - access("T"."ID_TICKET"="L"."ID_TICKET" AND "L"."ID_IMPUESTO"="LI"."ID_IMPUESTO")
   6 - access("CL"."ID_CLIENTE"="T"."ID_CLIENTE")
   7 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   8 - access("LI"."ID_INGRESO"="I"."ID_INGRESO")
   9 - access("PG"."ID_INGRESO"="I"."ID_INGRESO")
  10 - filter("PG"."TOTAL"<=800 AND "PG"."TOTAL">=500)
```

ANEXO 159

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 4 SELECT USANDO ÍNDICE Y SELECT USANDO ÍNDICE + JOIN 20 MILLONES DE DATOS.

```
Plan hash value: 2144962465
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		560K	91M		422K (1)	01:24:29
* 1	HASH JOIN		560K	91M		422K (1)	01:24:29
2	TABLE ACCESS FULL	PERSONA	10	310		2 (0)	00:00:01
* 3	HASH JOIN		560K	75M	3392K	422K (1)	01:24:29
4	TABLE ACCESS FULL	PRODUCTOS	78763	2461K		208 (1)	00:00:03
* 5	HASH JOIN		568K	59M	18M	418K (1)	01:23:46
* 6	HASH JOIN		189K	16M	12M	254K (1)	00:51:00
* 7	HASH JOIN		190K	10M	9864K	239K (1)	00:47:51
* 8	HASH JOIN		190K	7630K	7448K	196K (1)	00:39:13
* 9	HASH JOIN		190K	5210K	4288K	151K (1)	00:30:18
10	TABLE ACCESS BY INDEX ROWID	PAGOS	190K	2047K		107K (1)	00:21:26
* 11	INDEX RANGE SCAN	PAGOS_TOTAL	190K			410 (1)	00:00:05
12	TABLE ACCESS FULL	INGRESOS	20M	534M		16569 (2)	00:03:19
13	TABLE ACCESS FULL	LINEASIMPUESTO	20M	247M		20402 (2)	00:04:05
14	TABLE ACCESS FULL	TICKETS	20M	820M		16990 (2)	00:03:24
15	TABLE ACCESS FULL	CLIENTES	2240K	70M		10343 (1)	00:02:05
16	TABLE ACCESS FULL	LINEASTICKET	59M	1144M		71550 (2)	00:14:19

```
Predicate Information (identified by operation id):
---------------------------------------------------

   1 - access("T"."ID_PERSONA"="PER"."ID_PERSONA")
   3 - access("L"."ID_PRODUCTO"="P"."ID_PRODUCTO")
   5 - access("T"."ID_TICKET"="L"."ID_TICKET" AND "L"."ID_IMPUESTO"="LI"."ID_IMPUESTO")
   6 - access("CL"."ID_CLIENTE"="T"."ID_CLIENTE")
   7 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   8 - access("LI"."ID_INGRESO"="I"."ID_INGRESO")
   9 - access("PG"."ID_INGRESO"="I"."ID_INGRESO")
  11 - access("PG"."TOTAL">=500 AND "PG"."TOTAL"<=800)
```

ANEXO 160

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 4 SELECT USANDO SUBQUERY 20 MILLONES DE DATOS.

```
Plan hash value: 2112869973

-----------------------------------------------------------------------------------------------------------------------
| Id  | Operation                    | Name                   | Rows  | Bytes |TempSpc| Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                        |   571K|    34M|       |   269K  (1)| 00:53:59 |
|   1 |  TABLE ACCESS BY INDEX ROWID| CLIENTES               |     1 |    33 |       |     3   (0)| 00:00:01 |
|*  2 |   INDEX UNIQUE SCAN          | CLIENTES_ID_CLIENTE    |     1 |       |       |     2   (0)| 00:00:01 |
|   3 |  TABLE ACCESS BY INDEX ROWID| PRODUCTOS              |     1 |    32 |       |     2   (0)| 00:00:01 |
|*  4 |   INDEX UNIQUE SCAN          | PRODUCTOS_ID_PRODUCTO  |     1 |       |       |     1   (0)| 00:00:01 |
|*  5 |  TABLE ACCESS FULL           | LINEASIMPUESTO         |     1 |    13 |       | 20411   (2)| 00:04:05 |
|   6 |  TABLE ACCESS BY INDEX ROWID| PERSONA                |     1 |    31 |       |     1   (0)| 00:00:01 |
|*  7 |   INDEX UNIQUE SCAN          | PERSONA_ID_PERSONA     |     1 |       |       |     0   (0)| 00:00:01 |
|*  8 |  HASH JOIN                   |                        |   571K|    34M|    10M|   269K  (1)| 00:53:59 |
|*  9 |   HASH JOIN                  |                        |   190K|  8002K|  7448K|   106K  (1)| 00:21:18 |
|* 10 |    HASH JOIN                 |                        |   190K|  5210K|  4288K| 63424   (2)| 00:12:42 |
|* 11 |     TABLE ACCESS FULL        | PAGOS                  |   190K|  2047K|       | 19044   (2)| 00:03:49 |
|  12 |     TABLE ACCESS FULL        | INGRESOS               |    20M|   705M|       | 16569   (2)| 00:03:19 |
|  13 |    TABLE ACCESS FULL         | TICKETS                |    20M|   286M|       | 16990   (2)| 00:03:24 |
|  14 |   TABLE ACCESS FULL          | LINEASTICKET           |    59M|  1144M|       | 71550   (2)| 00:14:19 |
-----------------------------------------------------------------------------------------------------------------------

Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CL"."ID_CLIENTE"=:B1)
   4 - access("P"."ID_PRODUCTO"=:B1)
   5 - filter("LI"."ID_INGRESO"=:B1 AND "LI"."ID_IMPUESTO"=:B2)
   7 - access("PER"."ID_PERSONA"=:B1)
   8 - access("T"."ID_TICKET"="L"."ID_TICKET")
   9 - access("T"."ID_TICKET"="I"."ID_INGRESO")
  10 - access("PG"."ID_INGRESO"="I"."ID_INGRESO")
  11 - filter("PG"."TOTAL"<=800 AND "PG"."TOTAL">=500)
```

ANEXO 161

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 4 SELECT USANDO SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
Plan hash value: 2847663425
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		571K	41M		314K (1)	01:02:56
1	TABLE ACCESS BY INDEX ROWID	CLIENTES	1	33		3 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	CLIENTES_ID_CLIENTE	1			2 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
5	TABLE ACCESS BY INDEX ROWID	PERSONA	1	31		1 (0)	00:00:01
* 6	INDEX UNIQUE SCAN	PERSONA_ID_PERSONA	1			0 (0)	00:00:01
* 7	HASH JOIN		571K	41M	12M	314K (1)	01:02:56
* 8	HASH JOIN		190K	10M	9864K	151K (1)	00:30:15
* 9	HASH JOIN		190K	7630K	7448K	108K (1)	00:21:37
* 10	HASH JOIN		190K	5210K	4288K	63424 (2)	00:12:42
* 11	TABLE ACCESS FULL	PAGOS	190K	2047K		19044 (2)	00:03:49
12	TABLE ACCESS FULL	INGRESOS	20M	705M		16569 (2)	00:03:19
13	TABLE ACCESS FULL	LINEASIMPUESTO	20M	247M		20402 (2)	00:04:05
14	TABLE ACCESS FULL	TICKETS	20M	286M		16990 (2)	00:03:24
15	TABLE ACCESS FULL	LINEASTICKET	59M	1144M		71550 (2)	00:14:19

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CL"."ID_CLIENTE"=:B1)
   4 - access("P"."ID_PRODUCTO"=:B1)
   6 - access("PER"."ID_PERSONA"=:B1)
   7 - access("L"."ID_IMPUESTO"="LI"."ID_IMPUESTO" AND "L"."ID_TICKET"="T"."ID_TICKET")
   8 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   9 - access("LI"."ID_INGRESO"="I"."ID_INGRESO")
  10 - access("PG"."ID_INGRESO"="I"."ID_INGRESO")
  11 - filter("PG"."TOTAL"<=800 AND "PG"."TOTAL">=500)
```

ANEXO 162

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 4 SELECT USANDO ÍNDICE + SUBQUERY 20 MILLONES DE DATOS.

```
Plan hash value: 379305613
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		571K	34M		357K (1)	01:11:35
1	TABLE ACCESS BY INDEX ROWID	CLIENTES	1	33		3 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	CLIENTES_ID_CLIENTE	1			2 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
* 5	TABLE ACCESS FULL	LINEASIMPUESTO	1	13		20411 (2)	00:04:05
6	TABLE ACCESS BY INDEX ROWID	PERSONA	1	31		1 (0)	00:00:01
* 7	INDEX UNIQUE SCAN	PERSONA_ID_PERSONA	1			0 (0)	00:00:01
* 8	HASH JOIN		571K	34M	10M	357K (1)	01:11:35
* 9	HASH JOIN		190K	8002K	7448K	194K (1)	00:38:55
* 10	HASH JOIN		190K	5210K	4288K	151K (1)	00:30:18
11	TABLE ACCESS BY INDEX ROWID	PAGOS	190K	2047K		107K (1)	00:21:26
* 12	INDEX RANGE SCAN	PAGOS_TOTAL	190K			410 (1)	00:00:05
13	TABLE ACCESS FULL	INGRESOS	20M	705M		16569 (2)	00:03:19
14	TABLE ACCESS FULL	TICKETS	20M	286M		16990 (2)	00:03:24
15	TABLE ACCESS FULL	LINEASTICKET	59M	1144M		71550 (2)	00:14:19

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CL"."ID_CLIENTE"=:B1)
   4 - access("P"."ID_PRODUCTO"=:B1)
   5 - filter("LI"."ID_INGRESO"=:B1 AND "LI"."ID_IMPUESTO"=:B2)
   7 - access("PER"."ID_PERSONA"=:B1)
   8 - access("T"."ID_TICKET"="L"."ID_TICKET")
   9 - access("T"."ID_TICKET"="I"."ID_INGRESO")
  10 - access("PG"."ID_INGRESO"="I"."ID_INGRESO")
  12 - access("PG"."TOTAL">=500 AND "PG"."TOTAL"<=800)
```

ANEXO 163

PLAN DE EJECUCIÓN EN ORACLE 11G CONSULTA 4 SELECT USANDO ÍNDICE + SUBQUERY + JOIN 20 MILLONES DE DATOS.

```
Plan hash value: 2804176977
```

Id	Operation	Name	Rows	Bytes	TempSpc	Cost (%CPU)	Time
0	SELECT STATEMENT		571K	41M		402K (1)	01:20:33
1	TABLE ACCESS BY INDEX ROWID	CLIENTES	1	33		3 (0)	00:00:01
* 2	INDEX UNIQUE SCAN	CLIENTES_ID_CLIENTE	1			2 (0)	00:00:01
3	TABLE ACCESS BY INDEX ROWID	PRODUCTOS	1	32		2 (0)	00:00:01
* 4	INDEX UNIQUE SCAN	PRODUCTOS_ID_PRODUCTO	1			1 (0)	00:00:01
5	TABLE ACCESS BY INDEX ROWID	PERSONA	1	31		1 (0)	00:00:01
* 6	INDEX UNIQUE SCAN	PERSONA_ID_PERSONA	1			0 (0)	00:00:01
* 7	HASH JOIN		571K	41M	12M	402K (1)	01:20:33
* 8	HASH JOIN		190K	10M	9864K	239K (1)	00:47:51
* 9	HASH JOIN		190K	7630K	7448K	196K (1)	00:39:13
* 10	HASH JOIN		190K	5210K	4288K	151K (1)	00:30:18
11	TABLE ACCESS BY INDEX ROWID	PAGOS	190K	2047K		107K (1)	00:21:26
* 12	INDEX RANGE SCAN	PAGOS_TOTAL	190K			410 (1)	00:00:05
13	TABLE ACCESS FULL	INGRESOS	20M	705M		16569 (2)	00:03:19
14	TABLE ACCESS FULL	LINEASIMPUESTO	20M	247M		20402 (2)	00:04:05
15	TABLE ACCESS FULL	TICKETS	20M	286M		16990 (2)	00:03:24
16	TABLE ACCESS FULL	LINEASTICKET	59M	1144M		71550 (2)	00:14:19

```
Predicate Information (identified by operation id):
---------------------------------------------------

   2 - access("CL"."ID_CLIENTE"=:B1)
   4 - access("P"."ID_PRODUCTO"=:B1)
   6 - access("PER"."ID_PERSONA"=:B1)
   7 - access("L"."ID_IMPUESTO"="LI"."ID_IMPUESTO" AND "L"."ID_TICKET"="T"."ID_TICKET")
   8 - access("T"."ID_TICKET"="I"."ID_INGRESO")
   9 - access("LI"."ID_INGRESO"="I"."ID_INGRESO")
  10 - access("PG"."ID_INGRESO"="I"."ID_INGRESO")
  12 - access("PG"."TOTAL">=500 AND "PG"."TOTAL"<=800)
```

ANEXO 2

SCRIPTS DE LA CONSULTAS OPTIMIZADAS USANDO LA SENTENCIA SELECT

CONSULTA 1 SELECT SIMPLE

```
SELECT      l.id_ticket,
            l.id_producto,
            l.unidades,
            l.precio,
            (l.unidades * l.precio) * 0.12 as iva,
            p.nombre as descripcion_producto,
            a.descripcion as atributo,
            l.id_atributoconjuntoinstancia,
            im.monto
FROM        ingresos i, tickets t, lineasticket l, productos p, atributoconjuntoinstancia a,
            persona ps, roles r, lineasimpuesto im
WHERE       i.id_ingreso = t.id_ticket
            and t.id_ticket = l.id_ticket
            and p.id_producto = l.id_producto
            and a.id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia
            and ps.id_persona = t.id_persona
            and ps.id_rol = r.id_rol
            and r.nombre = 'Dependiente'
            and im.id_ingreso = i.id_ingreso
            and im.monto < 500
            and i.fecha between '20130101 00:00:00' and '20130131 00:00:00'
```

CONSULTA 1 SELECT USANDO ÍNDICE

```
CREATE INDEX ix_ingresos_fecha_entero ON ingresos(fecha_entero)
```

```
SELECT      l.id_ticket,
            l.id_producto,
            l.unidades,
            l.precio,
            (l.unidades * l.precio) * 0.12 as iva,
```

```
                p.nombre as descripcion_producto,
                a.descripcion as atributo,
                l.id_atributoconjuntoinstancia,
                im.monto
FROM            ingresos i, tickets t, lineasticket l, productos p, atributoconjuntoinstancia a,
                persona ps, roles r, lineasimpuesto im
WHERE           i.id_ingreso = t.id_ticket
                and t.id_ticket = l.id_ticket
                and p.id_producto = l.id_producto
                and a.id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia
                and ps.id_persona = t.id_persona
                and ps.id_rol = r.id_rol
                and r.nombre = 'Dependiente'
                and im.id_ingreso = i.id_ingreso
                and im.monto < 500
                and i.fecha between '20130101 00:00:00' and '20130131 00:00:00'
```

CONSULTA 1 SELECT USANDO SUBQUERY

```
SELECT          l.id_ticket,
                l.id_producto,
                l.unidades,
                l.precio,
                (l.unidades * l.precio) * 0.12 as iva,
                (SELECT p.nombre
                 FROM productos p
                WHERE p.id_producto = l.id_producto) as descripcion_producto,
                (SELECT a.descripcion
                 FROM atributoconjuntoinstancia a
                 WHERE a.id_atributoconjuntoinstancia =  l.id_atributoconjuntoinstancia) as
                atributo,
                l.id_atributoconjuntoinstancia
FROM            lineasticket l
WHERE           id_ticket in (
                        select id_ingreso
                        from lineasimpuesto
                        where id_ingreso in(select id_ingreso
                                            from ingresos
```

```
                                        where fecha between '20130101 00:00:00' and
                                        '20130131 00:00:00')
                        and monto < 500
                        )
and id_ticket in(
            select id_ticket
            from tickets
            where id_persona in (
                        select id_persona
                        from persona
                        where id_persona in(
                                    select id_persona
                                    from tickets
                                    where id_ticket in(
                                                select id_ingreso
                                                from ingresos
                                                where
                                                fecha between '20130101 00:00:00' and
                                                '20130131 00:00:00'
                                                            )
                                        )
                                and id_rol in (select id_rol
                                                from roles
                                                where nombre = 'Dependiente'
                                                )
                            ))
```

CONSULTA 1 SELECT USANDO JOIN

```
SELECT      l.id_ticket,
            l.id_producto,
            l.unidades,
            l.precio,
            (l.unidades * l.precio) * 0.12 as iva,
            p.nombre as descripcion_producto,
            a.descripcion as atributo,
            l.id_atributoconjuntoinstancia
FROM        ingresos i
            inner join
```

```
                tickets t on i.id_ingreso = t.id_ticket
        inner join
                lineasticket l on t.id_ticket = l.id_ticket
        inner join
                productos p on p.id_producto = l.id_producto
        inner join
                atributoconjuntoinstancia a on a.id_atributoconjuntoinstancia =
                l.id_atributoconjuntoinstancia
        inner join
                persona ps on ps.id_persona = t.id_persona
        inner join
                roles r on ps.id_rol = r.id_rol
        inner join
                lineasimpuesto im on im.id_ingreso = i.id_ingreso
WHERE           r.nombre = 'Dependiente'
        and im.monto < 500
        and i.fecha between '20130101 00:00:00' and '20130131 00:00:00'
```

CONSULTA 1 SELECT USANDO SUBQUERY + JOIN

```
SELECT      l.id_ticket,
            l.id_producto,
            l.unidades,
            l.precio,
            (l.unidades * l.precio) * 0.12 as iva,
            (select p.nombre
            from productos p
            where p.id_producto = l.id_producto ) as descripcion_producto,
            (select a.descripcion
            from atributoconjuntoinstancia a
            where a.id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia) as atributo,
            l.id_atributoconjuntoinstancia
FROM        ingresos i
            inner join
                tickets t on i.id_ingreso = t.id_ticket
            inner join
                lineasticket l on t.id_ticket = l.id_ticket
            inner join
                persona ps on ps.id_persona = t.id_persona
```

```
                inner join
                        lineasimpuesto im on im.id_ingreso = i.id_ingreso
WHERE           ps.id_rol in (select r.id_rol
                                from roles r
                                where ps.id_rol = r.id_rol
                                and r.nombre = 'Dependiente'
                            )
                and im.monto < 500
                and i.fecha between '20130101 00:00:00' and '20130131 00:00:00'
```

CONSULTA 1 SELECT USANDO ÍNDICE + JOIN

```
CREATE INDEX ingresos_fecha ON ingresos(fecha)

SELECT          l.id_ticket,
                l.id_producto,
                l.unidades,
                l.precio,
                (l.unidades * l.precio) * 0.12 as iva,
                p.nombre as descripcion_producto,
                a.descripcion as atributo,
                l.id_atributoconjuntoinstancia
FROM            ingresos i
                inner join
                        tickets t on i.id_ingreso = t.id_ticket
                inner join
                        lineasticket l on t.id_ticket = l.id_ticket
                inner join
                        productos p on p.id_producto = l.id_producto
                inner join
                        atributoconjuntoinstancia a on a.id_atributoconjuntoinstancia =
                        l.id_atributoconjuntoinstancia
                inner join
                        persona ps on ps.id_persona = t.id_persona
                inner join
                        roles r on ps.id_rol = r.id_rol
                inner join
                        lineasimpuesto im on im.id_ingreso = i.id_ingreso
WHERE           r.nombre = 'Dependiente'
```

```
                and im.monto < 500
                and i.fecha between '20130101 00:00:00' and '20130131 00:00:00'
```

CONSULTA 1 SELECT USANDO ÍNDICE + SUBQUERY

```
CREATE INDEX ingresos_fecha ON ingresos(fecha)

SELECT          t l.id_ticket,
                l.id_producto,
                l.unidades,
                l.precio,
                (l.unidades * l.precio) * 0.12 as iva,
                (select p.nombre
                from productos p
                where p.id_producto = l.id_producto) as descripcion_producto,
                (select a.descripcion
                from atributoconjuntoinstancia a
                where a.id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia
                ) as atributo,
                l.id_atributoconjuntoinstancia
FROM  lineasticket l
where   id_ticket in(select id_ingreso
                        from lineasimpuesto
                        where id_ingreso in
                                (select id_ingreso
                                 from ingresos
                                 where fecha between '20130101 00:00:00' and
                                '20130131 00:00:00')
                and monto < 500)
                and id_ticket in(select id_ticket
                                from tickets
                                where id_persona in
                                (select id_persona
                                from persona
                                where id_persona in
                                        (select id_persona
                                         from tickets
                                         where id_ticket in
                                                (select id_ingreso
```

```
                                    from ingresos
                                    where fecha between '20130101
                                    00:00:00' and '20130131 00:00:00'
                                     )
                            )
                    and id_rol in
                                    (select id_rol
                                    from roles
                                    where nombre = 'Dependiente'
                                )
            ))
```

CONSULTA 1 SELECT USANDO ÍNDICE + JOIN + SUBQUERY

```
CREATE INDEX ingresos_fecha ON ingresos(fecha)

SELECT      l.id_ticket,
            l.id_producto,
            l.unidades,
            l.precio,
            (l.unidades * l.precio) * 0.12 as iva,
            (select p.nombre
            from productos p
            where p.id_producto = l.id_producto) as descripcion_producto,
            (select a.descripcion
            from atributoconjuntoinstancia a
            where a.id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia) as  atributo,
            l.id_atributoconjuntoinstancia
FROM        ingresos i
            inner join
                tickets t on i.id_ingreso = t.id_ticket
            inner join
                lineasticket l on t.id_ticket = l.id_ticket
            inner join
                persona ps on ps.id_persona = t.id_persona
            inner join
                lineasimpuesto im on im.id_ingreso = i.id_ingreso
WHERE       ps.id_rol in (select r.id_rol
             from roles r
            where ps.id_rol = r.id_rol
```

```
                and r.nombre = 'Dependiente')
                and im.monto < 500
                and i.fecha between '20130101 00:00:00' and '20130131 00:00:00'
```

CONSULTA 2 SELECT SIMPLE

```
SELECT      cl.id_cliente,
            cl.nombres,
            cl.direccion,
            t.id_ticket,
            l.unidades,
            c.id_categoria,
            c.nombre,
            p.nombre as Descripcion_producto,
            p.precioventa,
            l.id_producto,
            l.id_atributoconjuntoinstancia,
            aci.descripcion
FROM        categorias c,productos p,atributoconjuntoinstancia aci,clientes cl,tickets t,
            lineasticket l
WHERE       t.id_ticket = l.id_ticket
            and p.id_producto=l.id_producto
            and aci.id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia
            and c.id_categoria=p.id_categoria
            and cl.id_cliente=t.id_cliente
            and c.nombre='FARMACEUTICOS'
            and aci.descripcion in ('ATRIB.CONJUNTO QUIFATEX PROP MERZ',
                                    'ATRIB.CONJUNTO NATURPHARMA',
                                    'ATRIB.CONJUNTO GRUNENTHAL-GENERICOS',
                                    'ATRIB.CONJUNTO VITAFARMA',
                                    'ATRIB.CONJUNTO QUIFATEX GENERICOS')
```

CONSULTA 2 SELECT USANDO ÍNDICE

```
CREATE  INDEX atribconjinst_descrip ON lineasticket (id_atributoconjuntoinstancia)
```

```
SELECT      cl.id_cliente,
            cl.nombres,
            cl.direccion,
```

```
                t.id_ticket,
                l.unidades,
                c.id_categoria,
                c.nombre,
                p.nombre as Descripcion_producto,
                p.precioventa,
                l.id_producto,
                l.id_atributoconjuntoinstancia,
                aci.descripcion
FROM            categorias c,productos p,atributoconjuntoinstancia aci,clientes cl,tickets
                t,lineasticket l
WHERE           t.id_ticket = l.id_ticket
                and p.id_producto=l.id_producto
                and aci.id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia
                and c.id_categoria=p.id_categoria
                and cl.id_cliente=t.id_cliente
                and c.nombre='FARMACEUTICOS'
                and aci.descripcion in ('ATRIB.CONJUNTO QUIFATEX PROP MERZ',
                                        'ATRIB.CONJUNTO NATURPHARMA',
                                        'ATRIB.CONJUNTO GRUNENTHAL-GENERICOS',
                                        'ATRIB.CONJUNTO VITAFARMA',
                                        'ATRIB.CONJUNTO QUIFATEX GENERICOS')
```

CONSULTA 2 SELECT USANDO SUBQUERY

```
SELECT *
FROM (

        SELECT      cl.id_cliente,
                    cl.nombres,
                    cl.direccion,
                    t.id_ticket,
                    l.unidades,
                    (select p.id_categoria
                    from productos p
                     where p.id_producto = l.id_producto ) as id_categoria,
                    (select c.nombre
                     from productos p, categorias c
                    where p.id_producto = l.id_producto
```

```
                and c.id_categoria = p.id_categoria) as nombre_categoria,
                (select p.nombre
                 from productos p
                where p.id_producto = l.id_producto ) as descripcion_poducto,
                (select p.precioventa
                from productos p
                where p.id_producto = l.id_producto ) as precioventa,
                l.id_producto,
                l.id_atributoconjuntoinstancia,
                (select a.descripcion
                from atributoconjuntoinstancia a
                where a.id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia) as
                atributo
        FROM    clientes cl, tickets t, lineasticket l
        WHERE   t.id_cliente = cl.id_cliente
                and l.id_ticket = t.id_ticket
                and l.id_atributoconjuntoinstancia in
                                (Select a.id_atributoconjuntoinstancia
                                from atributoconjuntoinstancia a
                                where descripcion in
                               ('ATRIB.CONJUNTO QUIFATEX PROP MERZ',
                                'ATRIB.CONJUNTO NATURPHARMA',
                                'ATRIB.CONJUNTO GRUNENTHAL-
                                GENERICOS',
                                 'ATRIB.CONJUNTO VITAFARMA',
                                'ATRIB.CONJUNTO QUIFATEX GENERICOS'))

) AS TB1 WHERE TB1.nombre_categoria = 'FARMACEUTICOS
```

CONSULTA 2 SELECT USANDO JOIN

```
SELECT      cl.id_cliente,
            cl.nombres,
            cl.direccion,
            t.id_ticket,
            l.unidades,
            c.id_categoria,
            c.nombre,
            p.nombre as Descripcion_producto,
```

```
                p.precioventa,
                l.id_producto,
                l.id_atributoconjuntoinstancia,
                aci.descripcion
FROM            tickets t
                inner join
                        lineasticket l on  t.id_ticket = l.id_ticket
                inner join
                        productos P on p.id_producto=l.id_producto
                inner join
                        categorias c on c.id_categoria=p.id_categoria
                inner join
                        atributoconjuntoinstancia aci on aci.id_atributoconjuntoinstancia =
                        l.id_atributoconjuntoinstancia
                inner join
                        clientes cl on cl.id_cliente=t.id_cliente
WHERE           c.nombre = 'FARMACEUTICOS'
                and aci.descripcion in
                                ('ATRIB.CONJUNTO QUIFATEX PROP MERZ',
                                 'ATRIB.CONJUNTO NATURPHARMA',
                                'ATRIB.CONJUNTO GRUNENTHAL-GENERICOS',
                                'ATRIB.CONJUNTO VITAFARMA',
                                'ATRIB.CONJUNTO QUIFATEX GENERICOS')
```

CONSULTA 2 SELECT USANDO SUBQUERY + JOIN

```
SELECT *
FROM (  SELECT        cl.id_cliente,
                      cl.nombres,
                      cl.direccion,
                      t.id_ticket,
                      l.unidades,
                      (select p.id_categoria
                        from productos p
                        where p.id_producto = l.id_producto ) as id_categoria,
                      (select c.nombre
                        from productos p, categorias c
                        where p.id_producto = l.id_producto
                        and c.id_categoria = p.id_categoria) as nombre_categoria,
                      (select p.nombre
```

```
                from productos p
                where p.id_producto = l.id_producto ) as descripcion_poducto,
              (select p.precioventa
                from productos p
                where p.id_producto = l.id_producto ) as precioventa,
              l.id_producto,
              l.id_atributoconjuntoinstancia,
              aci.descripcion as atributo
FROM          clientes cl
              inner join
                    tickets t on t.id_cliente = cl.id_cliente
              inner join
                    lineasticket l on l.id_ticket = t.id_ticket
              inner join
                    atributoconjuntoinstancia aci ON aci.id_atributoconjuntoinstancia
                    = l.id_atributoconjuntoinstancia
              and aci.descripcion in
                                  ('ATRIB.CONJUNTO QUIFATEX PROP MERZ',
                                  'ATRIB.CONJUNTO NATURPHARMA',
                                  'ATRIB.CONJUNTO GRUNENTHAL-
GENERICOS',
                                  'ATRIB.CONJUNTO VITAFARMA',
                                  'ATRIB.CONJUNTO QUIFATEX GENERICOS')
)AS TB1WHERE TB1.nombre_categoria = 'FARMACEUTICOS'
```

CONSULTA 2 SELECT USANDO ÍNDICE + JOIN

```
CREATE  INDEX atribconjinst_descrip ON lineasticket (id_atributoconjuntoinstancia)*/

SELECT        cl.id_cliente,
              cl.nombres,
              cl.direccion,
              t.id_ticket,
              l.unidades,
              c.id_categoria,
              c.nombre,
              p.nombre as Descripcion_producto,
              p.precioventa,
              l.id_producto,
```

```
                l.id_atributoconjuntoinstancia,
                aci.descripcion
FROM            tickets t
                inner join
                        lineasticket l on  t.id_ticket = l.id_ticket
                inner join
                        productos P on p.id_producto=l.id_producto
                inner join
                        categorias c on c.id_categoria=p.id_categoria
                inner join
                        atributoconjuntoinstancia aci on aci.id_atributoconjuntoinstancia =
                        l.id_atributoconjuntoinstancia
                inner join
                        clientes cl on cl.id_cliente=t.id_cliente
WHERE                   c.nombre = 'FARMACEUTICOS'
                and aci.descripcion in
                                ('ATRIB.CONJUNTO QUIFATEX PROP MERZ',
                                'ATRIB.CONJUNTO NATURPHARMA',
                                'ATRIB.CONJUNTO GRUNENTHAL-GENERICOS',
                                'ATRIB.CONJUNTO VITAFARMA',
                                'ATRIB.CONJUNTO QUIFATEX GENERICOS')
```

CONSULTA 2 SELECT USANDO ÍNDICE + SUBQUERY

```
CREATE INDEX atribconjinst_descrip ON lineasticket (id_atributoconjuntoinstancia)

SELECT *
FROM (          SELECT
                        cl.id_cliente,
                        cl.nombres,
                        cl.direccion,
                        t.id_ticket,
                        l.unidades,
                        (select p.id_categoria
                          from productos p
                          where p.id_producto = l.id_producto ) as id_categoria,
                        (select c.nombre
                          from productos p, categorias c
                          where p.id_producto = l.id_producto
                          and c.id_categoria = p.id_categoria) as nombre_categoria,
```

```
                    (select p.nombre
                      from productos p
                      where p.id_producto = l.id_producto ) as descripcion_poducto,
                    (select p.precioventa
                      from productos p
                      where p.id_producto = l.id_producto ) as precioventa,
                    l.id_producto,
                    l.id_atributoconjuntoinstancia,
                    (select a.descripcion
                      from atributoconjuntoinstancia a
                    where a.id_atributoconjuntoinstancia = l.id_atributoconjuntoinstancia) as
                    atributo
          FROM      clientes cl, tickets t, lineasticket l
          WHERE     t.id_cliente = cl.id_cliente
                    and  l.id_ticket = t.id_ticket
                    and l.id_atributoconjuntoinstancia in(Select
                                                            a.id_atributoconjuntoinstancia
                                                        from atributoconjuntoinstancia a
                                                        where descripcion in
                                                      ('ATRIB.CONJUNTO QUIFATEX PROP
                                                      MERZ',
                                                    'ATRIB.CONJUNTO NATURPHARMA',
                                                      'ATRIB.CONJUNTO GRUNENTHAL-
                                                      GENERICOS',
                                                      'ATRIB.CONJUNTO VITAFARMA',
                                                      'ATRIB.CONJUNTO QUIFATEX
                                                      GENERICOS')
                                                     )
) TB1 WHERE TB1.nombre_categoria = 'FARMACEUTICOS';
```

CONSULTA 2 SELECT USANDO ÍNDICE + SUBQUERY + JOIN

```
CREATE INDEX atribconjinst_descrip ON lineasticket (id_atributoconjuntoinstancia)

SELECT *
FROM (          SELECT cl.id_cliente,
                       cl.nombres,
                       cl.direccion,
                       t.id_ticket,
```

```
                l.unidades,
                (select p.id_categoria
                  from productos p
                  where p.id_producto = l.id_producto ) as id_categoria,
                (select c.nombre
                  from productos p, categorias c
                  where p.id_producto = l.id_producto
                  and c.id_categoria = p.id_categoria) as nombre_categoria,
                (select p.nombre
                  from productos p
                  where p.id_producto = l.id_producto ) as descripcion_poducto,
                (select p.precioventa
                  from productos p
                  where p.id_producto = l.id_producto ) as precioventa,
                l.id_producto,
                l.id_atributoconjuntoinstancia,
                aci.descripcion as atributo
        FROM     clientes cl
                inner join
                        tickets t on t.id_cliente = cl.id_cliente
                inner join
                        lineasticket l on l.id_ticket = t.id_ticket
                inner join
                        atributoconjuntoinstancia aci ON aci.id_atributoconjuntoinstancia
                        = l.id_atributoconjuntoinstancia
                and aci.descripcion in('ATRIB.CONJUNTO QUIFATEX PROP MERZ',
                                                'ATRIB.CONJUNTO NATURPHARMA',
                                                'ATRIB.CONJUNTO GRUNENTHAL-
                                                GENERICOS',
                                                'ATRIB.CONJUNTO VITAFARMA',
                                                'ATRIB.CONJUNTO QUIFATEX
GENERICOS') ) AS TB1 WHERE TB1.nombre_categoria = 'FARMACEUTICOS';
```

CONSULTA 3 SELECT SIMPLE

```
SELECT          l.id_producto,
                p.nombre,
                l.precio,
                sum(l.unidades)as unidades_vendidas,
```

```
                c.id_categoria,
                c.nombre as categoria
FROM            tickets t,ingresos i,lineasticket l,productos p,categorias c
WHERE           t.id_ticket = i.id_ingreso
                and t.id_ticket = l.id_ticket
                and p.id_producto = l.id_producto
                and c.id_categoria = p.id_categoria
                and i.fecha between '20130501' and '20130831'
group by        l.id_producto,
                p.nombre,
                l.precio,
                c.id_categoria,
                c.nombre
having          sum(l.unidades)between 600 and 1000
```

CONSULTA 3 SELECT USANDO ÍNDICE

```
CREATE INDEX ingresos_fecha ON ingresos(fecha)

SELECT          l.id_producto,
                p.nombre,
                l.precio,
                sum(l.unidades)as unidades_vendidas,
                c.id_categoria,
                c.nombre as categoria
FROM            tickets t,ingresos i,lineasticket l,productos p,categorias c
WHERE           t.id_ticket = i.id_ingreso
                and t.id_ticket = l.id_ticket
                and p.id_producto = l.id_producto
                and c.id_categoria = p.id_categoria
                and i.fecha between '20130501' and '20130831'
group by        l.id_producto,
                p.nombre,
                l.precio,
                c.id_categoria,
                c.nombre
having          sum(l.unidades)between 600 and 1000
```

CONSULTA 3 SELECT USANDO SUBQUERY

```
SELECT      (select p.id_categoria
            from productos p
            where p.id_producto = l.id_producto ) as id_categoria,
            (select c.nombre
            from productos p,categorias c
            where p.id_producto = l.id_producto
            and c.id_categoria =  p.id_categoria) as nombre_categoria,
            l.id_producto,
            (select p.nombre
            from productos p
            where p.id_producto = l.id_producto ) as nombre,
            l.precio,
            sum(l.unidades)as unidades_vendidas
FROM        lineasticket l
WHERE             l.id_ticket in (select id_ticket
                        from tickets
                        where id_ticket in (select id_ingreso
                                            from ingresos
                                            where fecha between '20130501' and
'20130831')
                                           )
group by    l.id_producto,
            l.precio
having      sum(l.unidades)between 600 and 1000
```

CONSULTA 3 SELECT USANDO JOIN

```
SELECT      l.id_producto,
            p.nombre,
            l.precio,
            sum(l.unidades)as unidades_vendidas,
            c.id_categoria,
            c.nombre as categoria
FROM        tickets t
            inner join
                  ingresos i on t.id_ticket = i.id_ingreso
            inner join
                  lineasticket l on t.id_ticket = l.id_ticket
```

```
                inner join
                        productos p on p.id_producto = l.id_producto
                inner join
                        categorias c on c.id_categoria = p.id_categoria
WHERE           i.fecha between '20130501' and '20130831'
group by        l.id_producto,
                p.nombre,
                l.precio,
                c.id_categoria,
                c.nombre
having          sum(l.unidades)between 600 and 1000
```

CONSULTA 3 SELECT USANDO SUBQUERY + JOIN

```
SELECT          l.id_producto,
                p.nombre,
                l.precio,
                sum(l.unidades)as unidades_vendidas,
                (select p.id_categoria
                from productos p
                where p.id_producto = l.id_producto
                ) as id_categoria,
                (select c.nombre
                from productos p inner join categorias c
                on p.id_producto = l.id_producto
                and c.id_categoria =  p.id_categoria
                ) as nombre_categoria
FROM            tickets t
                inner join
                lineasticket l on t.id_ticket = l.id_ticket
                inner join
                        productos p on p.id_producto = l.id_producto
                inner join
                        ingresos i on t.id_ticket = i.id_ingreso
                        and i.fecha between '20130501' and '20130831'
group by        p.nombre,
                l.id_producto,
                l.precio
having          sum(l.unidades)between 600 and 1000
```

CONSULTA 3 SELECT USANDO ÍNDICE + JOIN

```
CREATE INDEX ingresos_fecha ON ingresos(fecha)

SELECT      l.id_producto,
            p.nombre,
            l.precio,
            sum(l.unidades)as unidades_vendidas,
            c.id_categoria,
            c.nombre as categoria
FROM        tickets t
            inner join
                  ingresos i on t.id_ticket = i.id_ingreso
            inner join
                  lineasticket l on t.id_ticket = l.id_ticket
            inner join
                  productos p on p.id_producto = l.id_producto
            inner join
                  categorias c on c.id_categoria = p.id_categoria
            where i.fecha between '20130501' and '20130831'
group by    l.id_producto,
            p.nombre,
            l.precio,
            c.id_categoria,
            c.nombre
having      sum(l.unidades)between 600 and 1000
```

CONSULTA 3 SELECT USANDO ÍNDICE + SUBQUERY

```
CREATE INDEX ingresos_fecha ON ingresos(fecha)

SELECT      (select p.nombre
            from productos p
            where p.id_producto = l.id_producto) as nombre,
            l.precio,
            sum(l.unidades)as unidades_vendidas,
            (select p.id_categoria
            from productos p
            where p.id_producto = l.id_producto) as id_categoria,
```

```
                (select c.nombre
                from productos p,categorias c
                where p.id_producto = l.id_producto
                and c.id_categoria =  p.id_categoria) as nombre_categoria,
                l.id_producto
FROM            lineasticket l
where           l.id_ticket in (select id_ticket
                                from tickets
                                where id_ticket in(select id_ingreso
                                from ingresos
                                where fecha between '20130501' and '20130831')
                               )
group by        l.id_producto,
                l.precio
having          sum(l.unidades)between 600 and 1000
```

CONSULTA 3 SELECT USANDO ÍNDICE + JOIN + SUBQUERY

```
CREATE INDEX ingresos_fecha ON ingresos(fecha)

SELECT          l.id_producto,
                p.nombre,
                l.precio,
                sum(l.unidades)as unidades_vendidas,
                (select p.id_categoria
                from productos p
                where p.id_producto = l.id_producto) as id_categoria,
                (select c.nombre
                from productos p inner join categorias c
                on p.id_producto = l.id_producto
                and c.id_categoria =  p.id_categoria) as nombre_categoria
FROM            tickets t
                inner join
                lineasticket l on t.id_ticket = l.id_ticket
                inner join
                productos p on p.id_producto = l.id_producto

WHERE           l.id_ticket in (select id_ticket
                                from tickets
```

```
                    where id_ticket in(select id_ingreso
                                       from ingresos
                                       where fecha between '20130501' and
                                       '20130831')
                    )
group by    p.nombre,
            l.id_producto,
            l.precio
having      sum(l.unidades)between 600 and 1000
```

CONSULTA 4 SELECT SIMPLE

```
SELECT      t.id_ticket,
            cl.nombres as cliente,
            p.nombre as descripcion_producto,
            l.precio,
            l.unidades,
            li.monto as iva,
            pg.total as importe_total,
            i.fecha,
            per.nombre as vendedor
FROM         tickets t, clientes cl, lineasticket l, productos p,
            lineasimpuesto li,ingresos i, pagos pg, persona per
WHERE       t.id_ticket=l.id_ticket
            and t.id_persona=per.id_persona
            and t.id_ticket=i.id_ingreso
            and cl.id_cliente=t.id_cliente
            and l.id_producto=p.id_producto
            and l.id_impuesto=li.id_impuesto
            and li.id_ingreso=i.id_ingreso
            and pg.id_ingreso=i.id_ingreso
            and pg.total between 500 and 800
```

CONSULTA 4 SELECT USANDO ÍNDICE

```
CREATE INDEX pagos_total ON pagos (total)

SELECT      t.id_ticket,
```

```
                cl.nombres as cliente,
                p.nombre as descripcion_producto,
                l.precio,
                l.unidades,
                li.monto as iva,
                pg.total as importe_total,
                i.fecha,
                per.nombre as vendedor
FROM            tickets t, clientes cl, lineasticket l, productos p,
                lineasimpuesto li,ingresos i, pagos pg, persona per
WHERE           t.id_ticket=l.id_ticket
                and t.id_persona=per.id_persona
                and t.id_ticket=i.id_ingreso
                and cl.id_cliente=t.id_cliente
                and l.id_producto=p.id_producto
                and l.id_impuesto=li.id_impuesto
                and li.id_ingreso=i.id_ingreso
                and pg.id_ingreso=i.id_ingreso
                and pg.total between 500 and 800
```

CONSULTA 4 SELECT USANDO SUBQUERY

```
SELECT *
FROM (
                SELECT
                        t.id_ticket,
                        (SELECT cl.nombres
                        FROM clientes cl
                        WHERE cl.id_cliente=t.id_cliente)as cliente,
                        (SELECT p.nombre
                        FROM productos p
                        WHERE p.id_producto=l.id_producto)as descripcion_producto,
                        l.precio,
                        l.unidades,
                        (SELECT li.monto
                        FROM lineasimpuesto li
                        WHERE li.id_ingreso=i.id_ingreso
                        and l.id_impuesto=li.id_impuesto)as iva,
                        pg.total,
```

```
        i.fecha,
        (SELECT per.nombre
        FROM persona per
        WHERE per.id_persona=t.id_persona)as vendedor
    FROM   tickets t, ingresos i, lineasticket l,pagos pg
    WHERE t.id_ticket=l.id_ticket
        and t.id_ticket=i.id_ingreso
        and pg.id_ingreso=i.id_ingreso
        and pg.total between 500 and 800
)AS TB1
```

CONSULTA 4 SELECT USANDO JOIN

```
SELECT      t.id_ticket,
            cl.nombres as cliente,
            p.nombre as descripcion_producto,
            l.precio,
            l.unidades,
            li.monto as iva,
            pg.total as importe_total,
            i.fecha,
            per.nombre as vendedor
FROM        lineasticket l
            inner join
                tickets t on t.id_ticket=l.id_ticket
            inner join
                persona per on t.id_persona=per.id_persona
            inner join
                ingresos i on t.id_ticket=i.id_ingreso
            inner join
                clientes cl on cl.id_cliente=t.id_cliente
            inner join
                productos p on l.id_producto=p.id_producto
            inner join
                lineasimpuesto li on li.id_ingreso=i.id_ingreso
                and  l.id_impuesto=li.id_impuesto
            inner join
                pagos pg on pg.id_ingreso=i.id_ingreso
                and pg.total between 500 and 800
```

CONSULTA 4 SELECT USANDO SUBQUERY + JOIN

```
SELECT      t.id_ticket,
            (SELECT cl.nombres
            FROM clientes cl
            WHERE cl.id_cliente=t.id_cliente)as cliente,
            (SELECT p.nombre
            FROM productos p
            WHERE p.id_producto=l.id_producto)as descripcion_producto,
            l.precio,
            l.unidades,
            li.monto,
            pg.total,
            i.fecha,
            (SELECT per.nombre
            FROM persona per
            WHERE per.id_persona=t.id_persona)as vendedor
FROM        lineasticket l
            inner join
                  tickets t on t.id_ticket=l.id_ticket
            inner join
                  ingresos i on t.id_ticket=i.id_ingreso
            inner join
                  lineasimpuesto li on li.id_ingreso=i.id_ingreso
                  and  l.id_impuesto=li.id_impuesto
            inner join
                  pagos pg on pg.id_ingreso=i.id_ingreso
                  and pg.total between 500 and 800
```

CONSULTA 4 SELECT USANDO ÍNDICE + JOIN

```
CREATE  INDEX pagos_total ON pagos (total)

SELECT
            t.id_ticket,
            cl.nombres as cliente,
            p.nombre as descripcion_producto,
            l.precio,
            l.unidades,
```

```
                li.monto as iva,
                pg.total as importe_total,
                i.fecha,
                per.nombre as vendedor
FROM            lineasticket l
                inner join
                        tickets t on t.id_ticket=l.id_ticket
                inner join
                        persona per on t.id_persona=per.id_persona
                inner join
                        ingresos i on t.id_ticket=i.id_ingreso
                inner join
                        clientes cl on cl.id_cliente=t.id_cliente
                inner join
                         productos p on l.id_producto=p.id_producto
                inner join
                         lineasimpuesto li on  li.id_impuesto=l.id_impuesto
                        and  li.id_ingreso=i.id_ingreso
                inner join
                        pagos pg on pg.id_ingreso=i.id_ingreso
                        and pg.total between 500 and 800
```

CONSULTA 4 SELECT USANDO ÍNDICE + SUBQUERY

```
CREATE  INDEX pagos_totalON pagos (total)
SELECT *
FROM (
                SELECT t.id_ticket,
                        (SELECT cl.nombres
                        FROM clientes cl
                        WHERE cl.id_cliente=t.id_cliente)as cliente,
                        (SELECT p.nombre
                        FROM productos p
                        WHERE p.id_producto=l.id_producto)as descripcion_producto,
                        l.precio,
                        l.unidades,
                        (SELECT li.monto
                        FROM lineasimpuesto li
                        WHERE li.id_ingreso=i.id_ingreso
```

```
                and l.id_impuesto=li.id_impuesto)as iva,
                 pg.total,
                i.fecha,
                (SELECT per.nombre
                FROM persona per
                WHERE per.id_persona=t.id_persona)as vendedor
        FROM  tickets t, ingresos i, lineasticket l,pagos pg
        WHERE t.id_ticket=l.id_ticket
                and t.id_ticket=i.id_ingreso
                and pg.id_ingreso=i.id_ingreso
                and pg.total between 500 and 800
) as TB1
```

CONSULTA 4 SELECT USANDO ÍNDICE + JOIN + SUBQUERY

```
CREATE INDEX pagos_total ON pagos (total)

SELECT      t.id_ticket,
            (SELECT cl.nombres
            FROM clientes cl
            WHERE cl.id_cliente=t.id_cliente)as cliente,
            (SELECT p.nombre
            FROM productos p
            WHERE p.id_producto=l.id_producto)as descripcion_producto,
            l.precio,
            l.unidades,
            li.monto,
            pg.total,
            i.fecha,
            (SELECT per.nombre
            FROM persona per
            WHERE per.id_persona=t.id_persona)as vendedor
FROM              lineasticket l
            inner join
                  tickets t on t.id_ticket=l.id_ticket
            inner join
                  ingresos i on t.id_ticket=i.id_ingreso
            inner join
                  lineasimpuesto li on li.id_ingreso=i.id_ingreso
```

```
        and  l.id_impuesto=li.id_impuesto
inner join
        pagos pg on pg.id_ingreso=i.id_ingreso
        and pg.total between 500 and 800
```

ANEXO 164

MODELO DE OPTIMIZACIÓN

SENCILLA	SELECT (especificar campos necesarios) en lugar de * FROM (orden de las tablas con menor número de registros a mayor) alias de tabla WHERE (campos sin indexar)	Al realizar un SELECT simple se debe seleccionar solamente campos que necesite ya que el DBMS primero lee la estructura de la tabla antes de ejecutar la sentencia, también es importante usar alias de ésta manera ahorras al DBMS el tiempo de localizar a cual tabla pertenece el campo.
INDICE	**CREATE INDEX nombre_indice ON nombre_tabla(campo_tabla)** SELECT (seleccionar campos necesarios) FROM (orden de las tablas con menor número de registros a mayor) alias de tabla WHERE (campos indexados sobre los cuales se realiza la búsqueda)	No se debe aplicar índices en campos que son utilizados en una función porque estas hacen que no se utilice eficientemente los índices.
SUBQUERY	SELECT campo1, (SELECT campo2 FROM tabla1 as tb1 WHERE tb1.campo clave= tb3.campoclave), (SELECT campo3 FROM tabla2 as tb2 WHERE tb2.campoclave=tb3.campoclave) sum(campo1) FROM tabla3 as tb3 GROUP BY campo1 HAVING sum (campo1)	Cuando en una consulta se utilice group by es muy óptimo realizar subquery en los campos de selección; esto permite extraer columnas que nos interese, sin tener que incluirla en el group by mejorando el rendimiento.
JOIN	SELECT (campos necesarios) FROM (tabla1 con su respectivo alias) INNER JOIN tabla2 (alias de tabla) ON tb1.campoclave = tb2.campoclave	Utilizar INNER JOIN permite seleccionar todas las filas de ambas tablas, siempre y cuando haya una coincidencia entre las columnas en el inner join hay que usar solo las tablas que se necesitan..

UNIVERSIDAD TÉCNICA ESTATAL DE QUEVEDO

FACULTAD DE CIENCIAS DE LA INGENIERÍA

F O R M A T O

DATOS PERSONALES								
CÉDULA	NOMBRES	APELLIDOS	PROVINCIA	CANTÓN	PARROQUIA	DIRECCIÓN	TELÉFONO	CORREO
1205476854	MARÍA GABRIELA	VARAS BELTRÁN	LOS RÍOS	VENTANAS	VENTANAS	CDLA. LAS PALMERAS	0999519787	gabyvarasb_90@hotmail.com

DATOS DEL EGRESADO			
TÍTULO A OBTENER	CARRERA	PARALELO	FECHA DE EGRESO
INGENIERA EN SISTEMAS	INGENIERÍA EN SISTEMAS	"A"	MARZO 26 del 2013

DATOS DE TESIS						
TEMA DE TESIS	RES. APROB. DIRECCIÓN DE TESIS	RES. FECHA DE SUSTENTACIÓN	FECHA DE SUSTENTACIÓN	CÉDULA DEL TUTOR	NOMBRE DEL TUTOR	FECHA DE INCORPORACIÓN
"CREACIÓN DE UN MODELO DE OPTIMIZACIÓN PARA LOS QUERY UTILIZANDO LA SENTENCIA SELECT DE SQL"	37-HCD-FCI-2013	011-HCD-FCI-2014	MARZO 21 del 2014	0301114906	ING. ARIOSTO VICUÑA	MARZO 21 del 2014

CERTIFICACIÓN DE REDACCIÓN

Yo, **Lcda. Marjorie Torres Bolaños** con C.I. **070182756-0**, Docente de la Facultad de Ciencias de la Ingeniería de la Universidad Técnica Estatal de Quevedo, certifico que he revisado la tesis de grado de la Egresada **Varas Beltrán María Gabriela**, con C.I. **120547685-4** previo a la obtención del título de Ingeniera en Sistemas, titulada **"CREACIÓN DE UN MODELO DE OPTIMIZACIÓN PARA LOS QUERY UTILIZANDO LA SENTENCIA SELECT DE SQL",** habiendo cumplido con la redacción y corrección ortográfica que se ha indicado.

_____

Lcda. Marjorie Torres Bolaños