Diana Karina Santibáñez Santoscoy

Optimización de consultas basadas en costos y en reglas para bases de datos relacionales

Mar-2010

Universidad Autónoma de Querétaro
Facultad de Informática

Optimización de consultas basadas en costos y en reglas para bases de datos relacionales

Tesis
Que como parte de los requisitos para obtener el grado de Maestro en

Ingeniería de Software Distribuido

Presenta

ISC. Diana Karina Santibáñez Santoscoy

Querétaro, Qro., Marzo de 2010.

Universidad Autónoma de Querétaro
Facultad de Informática
Maestría en Ingeniería de Software Distribuido

OPTIMIZACIÓN DE CONSULTAS BASADAS EN COSTOS Y EN REGLAS
PARA BASES DE DATOS RELACIONALES

TESIS

Que como parte de los requisitos para obtener el grado de

Maestro en Ingeniería de Software Distribuido

Presenta:
ISC. Diana Karina Santibáñez Santoscoy

Dirigido por:
MSI. Gerardo Rodríguez Rojano

SINODALES

MSI. Gerardo Rodríguez Rojano
Presidente
Firma

MISD. Juan Salvador Hernández Valerio
Secretario
Firma

MISD. Carlos Alberto Olmos Trejo
Vocal
Firma

M.C. Alberto Lamadrid Álvarez
Suplente
Firma

MISD. Jesús Armando Rincones
Suplente
Firma

M.C. Ruth Angélica Rico Hernández
Directora de la Facultad

Dr. Luis Gerardo Hernández Sandoval
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Marzo de 2010
México

RESUMEN

La presente investigación tiene como objetivos principales conocer la forma en que las sentencias SQL son procesadas por un manejador de base de datos relacional, investigar las características de los enfoques basados en costos y en reglas para la optimización de sentencias SQL señalando sus similitudes y diferencias, conocer las herramientas de optimización de sentencias SQL que acompañan a un manejador de bases de datos y establecer lineamientos a considerar para el diseño de consultas eficientes. Se realizó una investigación bibliográfica y electrónica sobre el procesamiento y optimización de consultas SQL, tratando la forma en que son implementadas las operaciones del álgebra relacional, su costo de ejecución, así como los enfoques para la selección del mejor plan de evaluación de una consulta. Se describe la forma en que el manejador de base de datos Oracle 10g realiza el procesamiento de consultas junto con las estructuras y componentes involucrados en este proceso. Se describe la herramienta AUTOTRACE de SQL*Plus, utilizada en la afinación de sentencias SQL. Se estudia el optimizador de Oracle en sus dos enfoques: basado en costos y basado en reglas, mencionando sus características distintivas, rutas de acceso disponibles y parámetros de la instancia que influyen en su comportamiento. Se refieren lineamientos que permiten identificar sentencias SQL problemáticas y desarrollar consultas SQL eficientes. Se trata la herramienta SQL Tuning Advisor empleada en la afinación de SQL automático en Oracle. Se diseñó un caso práctico que permite ejecutar un conjunto predefinido de veintidós consultas bajo ambos enfoques de optimización. Se compararon las estadísticas de ejecución de lecturas consistentes, lecturas físicas y ordenaciones en memoria entre el optimizador basado en costos y basado en reglas por cada consulta y conjunto de consultas. Se seleccionaron los casos críticos para realizar el análisis de los planes de ejecución correspondientes. Los resultados obtenidos mostraron un mejor desempeño para el optimizador basado en costos en las estadísticas analizadas por lo que se concluyó que este enfoque de optimización mejora el uso de recursos computacionales y el tiempo de respuesta durante la ejecución de consultas en bases de datos relacionales empleadas por aplicaciones OLTP.

(**Palabras clave**: optimización, optimizador, costos, reglas)

SUMMARY

The present research has as main objectives understanding the way SQL sentences are processed by a relational database management system, researching the features of the cost based and rule based approaches for tuning SQL sentences, indicating their likenesses and differences, understanding the tools for tuning SQL sentences that come with a database management system and defining guidelines to be considered for efficient query design. A bibliographic and electronic research about the processing and tuning of SQL queries was performed, treating the way on which relational algebra operations are implemented, their execution cost, as well as the approaches for selecting the best execution plan of a query. The way how Oracle 10g database system performs query processing as long as the structures and components involved in this process is described. SQL*Plus AUTOTRACE tool, used in tuning SQL sentences is described. The Oracle optimizer on its two approaches, cost based and rule based is examined, mentioning their distinctive features, available access paths and instance parameters that affect its behavior. Guidelines that allow identifying problematic SQL sentences and developing efficient SQL queries are referred. The SQL Tuning Advisor tool used in the Oracle automated SQL tuning is considered. A practical case that lets executing a predefined set of twenty two queries under both optimization approaches was designed. Execution statistics of consistent gets, physical reads and sorts in memory between the cost based and the rule based optimizers for each query and set of queries were compared. Critical cases were selected to perform the corresponding execution plan analysis. The results obtained showed a better performance for the cost based optimizer on the analyzed statistics therefore it was concluded that this optimization approach improves the usage of computational resources and the response time during query execution on relational databases used by OLTP applications.

(**Keywords**: tuning, optimizer, cost, rule)

DEDICATORIAS

Con mucho cariño dedico este trabajo a mi familia,
por darme siempre el apoyo y la motivación
para la realización de esta experiencia de conocimiento.

INDICE

INDICE DE CUADROS

INDICE DE FIGURAS

1 INTRODUCCIÓN

1.1 Antecedentes y justificación.

Con el objetivo de señalar la importancia y los aspectos relevantes del tema de optimización de consultas, se presentan en este apartado los antecedentes pertinentes, se comienza por la definición de una consulta, se describe el lenguaje SQL como el medio para recuperar información de una base de datos relacional, se muestra la tarea del optimizador de la base de datos de definir cómo obtener los datos solicitados de una forma rápida, mediante el procesamiento y optimización de la consulta respectiva. Se mencionan los enfoques de optimización basado en costos y basado en reglas disponibles en Oracle así como sus características fundamentales. Finalmente se muestra la responsabilidad que tiene el desarrollador de aplicaciones de conocer y llevar a cabo el proceso de afinación de consultas como parte de su trabajo habitual para garantizar la creación de aplicaciones de buen desempeño.

Una consulta es una solicitud no procedural de información de la base de datos, donde el usuario especifica qué datos se necesitan sin especificar cómo obtener esos datos. [1] Mediante un lenguaje de acceso de alto nivel como SQL, los usuarios de una base de datos relacional pueden acceder todos sus datos empresariales dinámicamente sin ningún conocimiento de cómo los datos básicos son realmente almacenados. El sistema de base de datos relacional debe aceptar una variedad diversa de consultas de entrada y convertirlas a un formato que eficientemente acceda a los datos almacenados, manteniendo el desempeño del sistema y procesamiento. [2]

Antes de que la base de datos pueda ejecutar una sentencia SQL debe determinar exactamente cómo procesará los datos y qué rutas de acceso y algoritmos utilizará. [3]

El procesamiento de consultas se refiere a las actividades implicadas en la extracción de datos de una base de datos. [4]

La optimización de consultas es el proceso por el cual el sistema manejador de bases de datos decide exactamente como se ejecutará una consulta. El trabajo

del optimizador de consultas es determinar el mejor plan de ejecución, la forma más rápida de obtener el resultado deseado. [3]

Esperar que un optimizador siempre encuentre la forma ideal de ejecutar cualquier sentencia es poco realista; ya que SQL es extenso y contiene muchos términos, además los usuarios pueden expresar una consulta de varias formas y el sistema puede usar varios tipos de rutas de acceso y algoritmos para acceder a los datos o producir un resultado particular. [3]

Antes de la introducción del optimizador basado en costos, Oracle utilizó el enfoque heurístico para la optimización de consultas donde los planes de ejecución eran generados en la base de ciertas heurísticas o reglas. Consideraciones como el tamaño de la tabla, variabilidad de los datos para las columnas, selectividad de índices o parámetros específicos de la instancia no eran tomadas en cuenta. Si los datos en las tablas eran inconsistentes con las reglas, entonces planes subóptimos podrían ser generados. [1]

La optimización basada en costos introducida en Oracle 7, cuantifica el consumo de recursos de una consulta para elegir entre los planes de ejecución alternativos y descarta todos excepto aquel que producirá el resultado requerido al menor costo global. Típicamente el costo de E/S, es decir el costo de transferir información de disco a la memoria principal, es el más significativo para procesar una consulta. [1]

El costo de E/S es directamente proporcional al número de registros a ser recuperados y manipulados y que constituyen la cardinalidad de la consulta. Para determinar la cardinalidad se requiere el uso de factores de selectividad, que son estimados sobre el número de registros que satisfacerán ciertos criterios en valores de columna. Una vez que la cardinalidad es conocida, el siguiente punto a considerar es el costo asociado con cada método de recuperación disponible. A medida que el optimizador evalúa el costo de cada plan, lo compara con el plan de mejor costo visto hasta el momento. Continúa haciendo esto hasta que obtiene el mejor de la variedad de planes que considera. [1]

Para habilitar el cálculo de costos de planes de ejecución, se requieren descripciones estadísticas detalladas de los datos relacionados con los objetos en la consulta. [1]

La diferencia entre los dos optimizadores es relativamente clara, el optimizador basado en costos elige la mejor ruta para las consultas, basado en lo que sabe sobre los datos y haciendo uso de características de la base de datos, mientras que el optimizador basado en reglas solo sigue reglas establecidas o heurísticas. [5]

Muchos desarrolladores falsamente asumen que su único objetivo es escribir sentencias SQL que entreguen los datos correctos. En realidad, formular el SQL es únicamente la mitad de su trabajo. Los negocios exitosos requieren siempre que los desarrolladores se aseguren que sus sentencias SQL acceden la base de datos en una manera óptima. [6]

El proceso de afinación de sentencias SQL presenta varios retos al desarrollador de aplicaciones. Primero requiere un alto nivel de experiencia en varias áreas complejas: optimización de consultas, diseño de acceso, diseño SQL. Segundo, es un proceso que consume tiempo porque cada sentencia es única y requiere ser tratada individualmente; además, el número de sentencias puede ser muy amplio. Tercero, requiere un conocimiento íntimo de estructuras del esquema y el modelo de uso de datos de la aplicación. Finalmente, la actividad de afinación de SQL es una tarea constante porque la carga de trabajo de SQL siempre está cambiando. También, los cambios en las estructuras de acceso a datos son muy posibles que causen cambios en los planes de ejecución, forzando al desarrollador de aplicaciones a empezar de nuevo. [7]

Entender el optimizador de consultas puede ayudar a los desarrolladores de aplicaciones a crear aplicaciones de buen desempeño. [3]

1.2 Objetivos generales de la investigación.

- Conocer la forma en que las sentencias SQL son procesadas por un manejador de base de datos relacional.

- Investigar los métodos existentes para la optimización de sentencias SQL.
- Investigar las características del criterio basado en costos y del criterio basado en reglas de Oracle.
- Determinar las similitudes y diferencias entre estos criterios.
- Conocer las herramientas de optimización de sentencias que acompañan a un manejador de bases de datos relacional.
- Establecer lineamientos a considerar para el diseño de consultas.

1.3 Hipótesis que sustenta el trabajo.

Con la aplicación de los criterios basados en costos y en reglas para la optimización de consultas se puede mejorar el tiempo de respuesta de una consulta y el uso de recursos computacionales.

1.4 Metodología de investigación.

Para la primera fase de la investigación se siguió un enfoque cualitativo describiendo las actividades involucradas en el procesamiento de consultas y su fase de optimización, la forma en que se implementan las operaciones del álgebra relacional y las estadísticas de base de datos que permiten estimar el tamaño del resultado de una operación y su costo. Explicando también las cualidades de los enfoques para la selección del mejor plan de evaluación, basado en costos y en reglas, materia de esta investigación.

Se continuó con un análisis descriptivo de los objetivos de afinación que varían dependiendo de las necesidades de la aplicación así como de los problemas comunes de desempeño. Se detallan las fases de procesamiento de sentencias SQL de Oracle 10g y los principales componentes del SGA del servidor que intervienen en este proceso, explicando los enfoques de optimización, basado en costos y basado en reglas, con sus rutas de acceso disponibles. Se describe la herramienta de afinación SQL*Plus AUTOTRACE y la función de afinación de SQL automático de Oracle 10g disponible con la herramienta SQL Tuning Advisor.

En la parte final se empleó un enfoque cuantitativo, analizando las estadísticas arrojadas por la herramienta SQL*Plus AUTOTRACE al ejecutar un conjunto predefinido de consultas mediante los enfoques de optimización basado en costos y basado en reglas de Oracle 10g. Se hicieron comparaciones al nivel de consulta y de conjunto de las estadísticas de lecturas consistentes, lecturas físicas y ordenaciones en memoria entre ambos optimizadores para poder determinar el optimizador que ofrece mejor rendimiento.

1.5 Organización del contenido.

El capítulo de Revisión de Literatura aborda las fases del procesamiento de una consulta SQL. Describe cuál es el costo más significativo para la evaluación de una consulta y la importancia de las estadísticas de base de datos para estimar el tamaño y costo de las operaciones del álgebra relacional.

Para las operaciones de selección, reunión y proyección del álgebra relacional, se mencionan las opciones para su implementación y costo estimado.

Después se indican los enfoques para la evaluación de expresiones del álgebra relacional, continuando con la descripción de la optimización de consultas y de los enfoques para la selección del mejor plan de evaluación de una consulta: la optimización basada en costos y basada en reglas.

Se tratan los objetivos de afinación de acuerdo al tipo de aplicación, On-Line Transaction Processing ó Decision Support Systems, así como los problemas comunes de desempeño en un sistema que provocan que no sea escalable. Se ubica la afinación de sentencias SQL como responsabilidad del desarrollador de aplicaciones.

También se describe el System Global Area, un área de memoria del servidor Oracle que almacena información de la base de datos y que contiene sub-áreas como el shared pool, el cual contiene áreas SQL compartidas utilizadas durante el procesamiento de sentencias SQL.

Se señalan los componentes principales y las fases para el procesamiento de una sentencia SQL en Oracle.

Se describe la herramienta AUTOTRACE de SQL*Plus, que se utiliza en la afinación de sentencias SQL. Se estudia el optimizador de Oracle en sus dos enfoques: basado en costos y basado en reglas. Para cada enfoque se mencionan sus características en cuanto a cómo eligen la manera más eficiente de ejecutar una sentencia SQL, sus rutas de acceso disponibles y parámetros de la instancia que influyen en su comportamiento.

Se menciona la forma manual y automática para reunir estadísticas en los objetos de la base de datos Oracle. Se describen los histogramas, que son estadísticas de columna que permiten conocer la distribución de los datos.

Se refieren lineamientos que permiten identificar sentencias SQL problemáticas y desarrollar sentencias SQL eficientes.

Se estudia la herramienta SQL Tuning Advisor empleada en la afinación de SQL automático en Oracle 10g.

2 REVISIÓN DE LITERATURA

2.1 Procesamiento de una consulta SQL.

Se refiere a las actividades implicadas en la extracción de datos de una base de datos. El procesamiento de consultas puede ser dividido en tres fases principales: descomposición, optimización, y evaluación. [4]

1) Descomposición. [8]

En esta fase se transforma una consulta de alto nivel en una consulta de álgebra relacional, y se verifica que la consulta es sintáctica y semánticamente correcta.

Las fases típicas de la descomposición son análisis, normalización, análisis semántico, simplificación y reestructuración de la consulta.

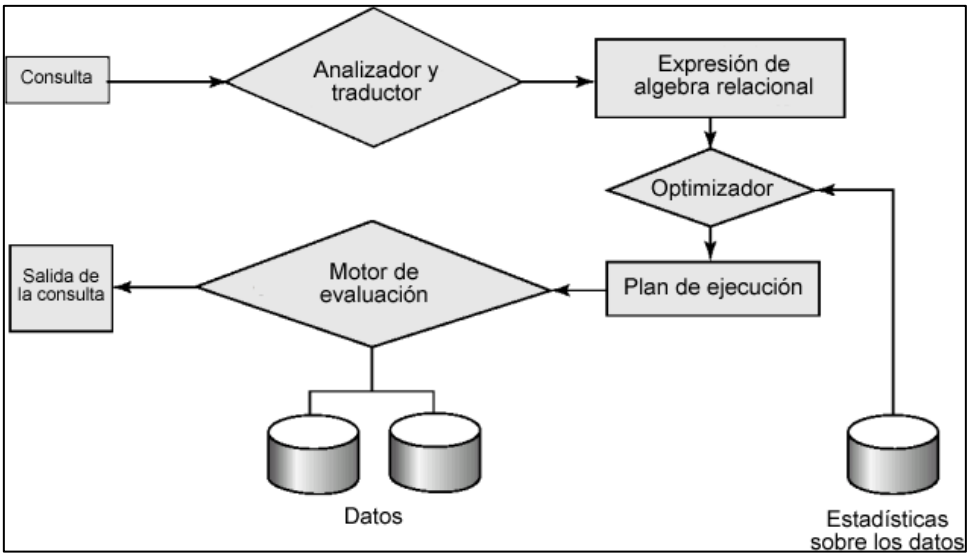

Figura 2.1 Diagrama general del procesamiento de una consulta SQL.

a) Análisis.

En esta etapa la consulta es analizada léxica y sintácticamente de forma similar al trabajo que realiza el analizador de un compilador. Se verifica que las relaciones y atributos especificados en la consulta están definidos en el catálogo del sistema. También se verifica que cualquier operación aplicada a los objetos de base de datos es apropiada para el tipo de objeto.

En la finalización de esta etapa, la consulta de alto nivel ha sido transformada en alguna representación interna que es más adecuada para el procesamiento. La forma interna típicamente elegida es alguna clase de árbol de consulta que se construye como sigue:

- Un nodo hoja es creado para cada relación base en la consulta.
- Un nodo interno es creado para cada relación intermedia producida por una operación de álgebra relacional.
- La raíz del árbol representa el resultado de la consulta.
- La secuencia de operaciones es dirigida desde las hojas hacia la raíz.

Este tipo de árbol de consulta es conocido como árbol de álgebra relacional.

b) Normalización.

En esta etapa se convierte la consulta en una forma normalizada que puede ser manipulada más fácilmente. El predicado, la cláusula WHERE, que puede ser arbitrariamente complejo, puede convertirse en una de dos formas aplicando unas cuantas reglas de transformación:

Forma Normal Conjuntiva.- Una secuencia de proposiciones que están conectadas con el operador $\wedge$ (AND). Cada término de la conjunción contiene uno o más términos conectados por el operador $\vee$ (OR). Una selección conjuntiva contiene únicamente aquellas tuplas que satisfacen todas las conjunciones.

Forma Normal Disyuntiva.- Una secuencia de proposiciones que están conectadas con el operador $\vee$ (OR). Cada término de la disyunción contiene uno o más términos conectados por el operador $\wedge$ (AND). Una selección disyuntiva contiene aquellas tuplas formadas por la unión de todas las tuplas que satisfacen las disyunciones.

c) Análisis Semántico.

El objetivo del análisis semántico es rechazar consultas normalizadas que son formuladas incorrectamente o contradictorias.

Una consulta es formulada incorrectamente si los componentes no contribuyen a la generación del resultado, lo cual puede ocurrir si están faltando algunas especificaciones de reunión.

Una consulta es contradictoria si su predicado no puede ser satisfecho por ninguna tupla.

d) Simplificación.

Los objetivos de la fase de simplificación son detectar calificaciones redundantes, eliminar subexpresiones comunes, y transformar la consulta a una forma semánticamente equivalente pero calculada más fácil y eficientemente. Típicamente, las restricciones de acceso, definiciones de vista, y restricciones de integridad son consideradas en esta fase, algunas de las cuales pueden introducir también redundancia.

e) Reestructuración de la Consulta.

En la etapa final de la descomposición de una consulta, la consulta es reestructurada para proporcionar una implementación más eficiente.

2) Optimización. Es la actividad de escoger una estrategia de ejecución eficiente para procesar una consulta. [8]

Cada consulta en SQL se puede traducir en una expresión del álgebra relacional de varias formas, es decir, existen muchas transformaciones equivalentes de la misma consulta de alto nivel. La representación de una consulta en el álgebra relacional especifica parcialmente cómo evaluar la consulta; ya que normalmente existen varias maneras de evaluar expresiones del álgebra relacional.

Un plan de ejecución de la consulta o plan de evaluación es una secuencia de operaciones del álgebra relacional con instrucciones que especifican como evaluar cada operación y que se puede utilizar para evaluar una consulta. [4]

Existen dos técnicas principales para la optimización de consultas, a pesar de que las dos estrategias son combinadas usualmente en la práctica. La primer técnica utiliza reglas heurísticas que ordenan las operaciones en una consulta. La otra técnica compara diferentes estrategias basadas en sus costos relativos y selecciona aquella que minimiza el uso de recursos. [8]

3) Evaluación. Una vez elegido el plan de la consulta se evalúa la misma con ese plan y se muestra el resultado de la consulta. [4]

2.2 Medidas del costo de una consulta.

El costo de evaluación de una consulta se puede expresar en términos de diferentes recursos: accesos a disco, tiempo de CPU y el costo de la comunicación en sistemas de bases de datos distribuidos o paralelos.

El tiempo de respuesta para un plan de evaluación de una consulta podría tener en cuenta todos estos costos y utilizarlos como buena medida del costo del plan.

Ya que el acceso a disco es lento comparado con el acceso a memoria, el acceso a disco tiende a ser el costo dominante en el procesamiento de una consulta para un DBMS centralizado. Por esto se considera el costo de los accesos a disco una medida razonable del costo del plan de evaluación de una consulta. [4]

Se mencionarán las diferentes opciones disponibles para implementar las operaciones del álgebra relacional, proporcionando la introducción de su implementación y costo estimado. Asumiendo lo siguiente: [8]

- Todas las transferencias de bloques tienen el mismo costo para simplificar los cálculos de costo de los accesos a disco.
- Las estimaciones de costo que se mencionan ignoran el costo de escribir el resultado final de una operación en disco.
- Los costos de todos los algoritmos que se consideran dependen del tamaño de la memoria intermedia en la memoria principal. En el mejor caso, todos los datos se pueden leer en las memorias intermedias. En el peor caso, la memoria intermedia puede contener sólo unos pocos bloques de datos, aproximadamente un bloque por relación.

2.3 Estadísticas de base de datos.

El éxito en la estimación del tamaño y costo de las operaciones intermedias del álgebra relacional depende de la cantidad y vigencia de la información estadística que se almacena en los catálogos de los sistemas de bases de datos. [8]

nTuplas(R)	Número de tuplas de la relación R, su cardinalidad.
bFactor(R)	Factor de bloqueo de la relación R. Número de tuplas de la relación R que caben en un bloque.
nBloques(R)	Número de bloques que contienen tuplas de la relación R. nBloques(R) = [nTuplas(R)/bFactor(R)] Se utiliza [*x*] para indicar que el resultado del cálculo se redondea al entero más pequeño que es mayor o igual a *x*.
tTupla(R)	Tamaño de cada tupla de la relación R en bytes.

Tabla 2.1 Estadísticas para cada relación base R.

$nDistinto_A(R)$	Número de valores distintos que aparecen en la relación R para el atributo A.	
$min_A(R)$, $max_A(R)$	El valor mínimo y máximo posibles para el atributo A en la relación R.	
$CS_A(R)$	La cardinalidad de la selección del atributo A en R. Esto es el número promedio de tuplas que satisfacen una condición de igualdad en el atributo A.	
Si se asume que los valores de A están uniformemente distribuidos en R, y que existe al menos un valor que satisface la condición, entonces:		
$CS_A(R)$ =	1	Si A es un atributo clave de R.
	$nTuplas(R)/nDistinto_A(R)$	En caso contrario.
	Para otras condiciones:	
	$nTuplas(R)*((max_A(R) – c)/(max_A(R) - min_A(R)))$	Para desigualdad (A > c).
	$nTuplas(R)*((c - min_A(R))/(max_A(R) - min_A(R)))$	Para desigualdad (A < c).
	$(nTuplas(R)/nDistinto_A(R))*n$	Para A en $\{c_1, c_2, ..., c_n\}$
	$CS_A(R) * CS_B(R)$	Para $(A \wedge B)$
	$CS_A(R) + CS_B(R) - CS_A(R) * CS_B(R)$	Para $(A \vee B)$

Tabla 2.2 Estadísticas para cada atributo A de la relación base R.

$nNiveles_A(I)$	El número de niveles en el índice I.
$nBloquesHoja_A(I)$	El número de bloques hoja en el índice I.

Tabla 2.3 Para cada índice multinivel I en el conjunto de atributos A.

Puede ser que las estadísticas utilizadas para escoger una estrategia de procesamiento de consultas no sean completamente exactas. Sin embargo, si no se producen demasiadas actualizaciones en los intervalos entre las actualizaciones de las estadísticas, éstas serán lo bastante precisas como para proporcionar una buena estimación de los costos relativos de los diferentes planes. [9]

2.4 Operación selección ($S = \sigma_P(R)$)

La operación selección en el álgebra relacional trabaja en una única relación R, y define una relación S conteniendo únicamente aquellas tuplas de R que satisfacen el predicado especificado. [8]

2.4.1 Cardinalidad de la operación selección.

La estimación del tamaño del resultado de una operación de selección depende del predicado de la selección. Si se asume que los valores de los atributos están uniformemente distribuidos dentro de su dominio y que los atributos son independientes, se tiene lo siguiente:

$\sigma_{A = a}(R) = CS_A(R)$

Para $\sigma_{A \leq v}(R)$ ó $\sigma_{A \geq v}(R)$ referirse a la Tabla 2.2 Estadísticas para cada atributo A de la relación base R.

La probabilidad o selectividad de que una tupla de la relación satisfaga la condición de selección θi es $CS_A(R)/nTuplas(R)$.

2.4.2 Algoritmos básicos.

Existen varias implementaciones diferentes para la operación Selección, dependiendo de la estructura del archivo en el cual la relación está almacenada, y en si lo(s) atributo(s) involucrados en el predicado han sido indexados o asociados. [8]

2.4.2.1 A1, búsqueda lineal.

Se explora cada bloque del archivo y se comprueban todos los registros para ver si satisfacen la condición de selección.

El costo de la búsqueda lineal es nBloques(R). Las selecciones [con condiciones de igualdad] sobre atributos clave tienen un costo medio de [nBloques(R)/2] asumiendo que las tuplas están uniformemente distribuidas sobre el archivo.

Este algoritmo se puede aplicar a cualquier archivo, sin importar la ordenación del archivo, la presencia de índices o la naturaleza de la operación selección.

2.4.2.2 A2, búsqueda binaria.

Si el archivo está ordenado según un atributo y la condición de selección es una comparación de igualdad en ese atributo, se puede utilizar una búsqueda binaria para localizar los registros que satisfacen la selección. El costo en este caso es $[\log_2(nBloques(R))]$.

Si la selección no es sobre un atributo clave [pero es el atributo sobre el cual el archivo está ordenado], más de un bloque puede contener los registros requeridos, y el costo de la lectura de los bloques extra se debe añadir a la estimación de costo. El costo para este caso es $[\log_2(nBloques(R))] + [CS_A(R)/bFactor(R)] - 1$.

2.4.2.3 Selecciones con índices.

Los algoritmos de búsqueda que usan un índice son:

2.4.2.3.1 A3, índice primario, igualdad en llave candidata.

Se puede utilizar el índice para recuperar el único registro que satisface la correspondiente condición de igualdad.

El costo en términos de operaciones de E/S [~~usando un árbol B^+~~] es $nNiveles_A(I) + 1$.

2.4.2.3.2 A4, índice primario, igualdad en atributo no clave.

La única diferencia del caso anterior es que puede ser necesario recuperar varios registros. Sin embargo, estos registros estarían en bloques consecutivos ya que el archivo se ordena según la clave de búsqueda.

El costo estimado es $nNiveles_A(I) + [CS_A(R)/bFactor(R)]$, donde el segundo término es un estimado del número de bloques que serán requeridos para almacenar el número de tuplas que satisfacen la condición de igualdad, la cual se ha estimado como $CS_A(R)$.

2.4.2.3.3 A5, índice secundario, igualdad.

Este algoritmo puede recuperar un solo registro si la clave de búsqueda es una clave candidata. Si no es así, se recuperarían varios registros.

Para este caso se asume que las tuplas están en diferentes bloques ya que el índice es secundario, así que el costo estimado se convierte en $nNiveles_A(I) + [CS_A(R)]$.

Selecciones con condiciones de comparación.

Considerando una selección de la forma $\sigma_{A \leq v}(R)$ [solo menor o igual ?] se puede implementar utilizando búsqueda lineal, binaria, o con índices de acuerdo a lo siguiente:

2.4.2.3.4 A6, índice primario, comparación.

Se utiliza primero el índice primario para encontrar la tupla que satisface el predicado A = v. Entonces las tuplas requeridas pueden encontrarse accediendo a todas las tuplas antes o después de la tupla encontrada.

El costo estimado es $nNiveles_A(I) + [nBloques(R)/2]$ ya que asumiendo una distribución uniforme, se espera que la mitad de las tuplas satisfagan la desigualdad.

2.4.2.3.5 A7, índice secundario, comparación.

Si el predicado involucra una condición de desigualdad en el atributo A, el cual proporciona un índice secundario B^+, entonces desde los nodos hoja del árbol, se pueden examinar las claves desde el valor más pequeño hasta x (para condiciones <, <=) o desde x hasta el valor máximo (para condiciones >, >=). Asumiendo una distribución uniforme se esperaría que la mitad de los bloques nodo hoja sean accedidos y, que a través del índice, la mitad de las tuplas sean accedidas.

El costo estimado es entonces $nNiveles_A(I) + [nBloquesHoja_A(I)/2 + nTuplas(R)/2]$.

A1, búsqueda lineal.	[nBloques(R)/2] para condiciones de igualdad en atributo clave. nBloques(R) de otra manera.
A2, búsqueda binaria.	$[\log_2(nBloques(R))]$ para condiciones de igualdad en atributo ordenado. $[\log_2(nBloques(R))] + [CS_A(R)/bFactor(R)] - 1$, de otra manera.
Selecciones con índices.	

A3, índice primario, igualdad en llave candidata.	$nNiveles_A(I) + 1$
A4, índice primario, igualdad en atributo no clave.	$nNiveles_A(I) + [CS_A(R)/bFactor(R)]$
A5, índice secundario, igualdad.	$nNiveles_A(I) + [CS_A(R)]$
Selecciones con condiciones de comparación.	
A6, índice primario, comparación.	$nNiveles_A(I) + [nBloques(R)/2]$
A7, índice secundario, comparación.	$nNiveles_A(I) + [nBloquesHoja_A(I)/2 + nTuplas(R)/2]$

Tabla 2.4 Resumen del costo estimado de operaciones de E/S de las estrategias para la operación Selección.

2.5 Operación reunión ($(T = R \triangleright\lhd_F S)$)

La operación reunión Theta define una relación que contiene las tuplas que satisfacen un predicado específico F del producto cartesiano de dos relaciones, R y S. El predicado F es de la forma R.a θ S.b, donde θ puede ser alguno de los operadores de comparación lógicos. Si el predicado contiene únicamente igualdad (=), la reunión es una equirreunión. Si la reunión involucra todos los atributos comunes de R y S, la reunión se conoce como reunión natural. [8]

2.5.1 Cardinalidad de la operación reunión.

La cardinalidad del producto Cartesiano de R y S, R x S, es simplemente nTuplas(R) * nTuplas(S).

Desgraciadamente es más difícil estimar la cardinalidad de cualquier reunión, ya que depende de la distribución de valores en los atributos de reunión. En el peor caso se sabe que la cardinalidad de la reunión no puede ser mayor que la cardinalidad del producto Cartesiano.

Asumiendo una distribución uniforme de valores en ambas relaciones se puede mejorar este estimado para equirreuniones con un predicado (R.A = S.B) de acuerdo a lo siguiente:

1. Si A es un atributo clave de R, entonces una tupla de S se puede reunir únicamente con una tupla de R. Por lo tanto, la cardinalidad de la equirreunión no puede ser más grande que la cardinalidad de S.

   $nTuplas(T) \leq nTuplas(S)$
2. De forma similar, si B es una clave de S, entonces:

   $nTuplas(T) \leq nTuplas(R)$

3. Si ni A ni B son claves, entonces se podría estimar la cardinalidad de la reunión como:

   nTuplas(T) = CS_A(R) * nTuplas(S)

   ó

   nTuplas(T) = CS_B(S) * nTuplas(R)

Para obtener el primer estimado, se usa el hecho de que para cualquier tupla *s* en S, se esperaría un promedio de CS_A(R) tuplas con un valor dado para el atributo A, y este número aparecería en la reunión. Multiplicando esto por el número de tuplas en S, se obtiene el primer estimado. Igualmente para el segundo estimado.

Si estas dos estimaciones son diferentes, probablemente haya tuplas pendientes que no participen en la reunión, por lo que la menor de las dos estimaciones posiblemente sea la más precisa.

Se muestran varios algoritmos para calcular la reunión de relaciones y para analizar sus costos asociados.

2.5.2 Algoritmos básicos.

2.5.2.1 Reunión en bucle anidado (Nested Loop).

Consiste básicamente de un par de bucles “for” anidados. La relación R se denomina la relación externa y S la relación interna de la reunión, ya que el bucle de R incluye al bucle de S.

Este algoritmo no necesita índices y puede utilizarse sin importar la condición de reunión.

El algoritmo de reunión en bucle anidado es costoso, ya que examina cada pareja de tuplas en las dos relaciones. En el peor de los casos la memoria intermedia solamente puede contener un bloque de cada relación necesitándose un total de nBloques(R) + nTuplas(R)*nBloques(S) accesos a bloques.

En el mejor de los casos, hay suficiente espacio para que las dos relaciones quepan en memoria, así que cada bloque se tendrá que leer solamente una vez; accediendo así a nBloques(R) + nBloques(S) bloques.

Si una de las relaciones cabe en memoria por completo, es útil emplearla como la relación más interna, ya que solo será necesario leer una vez la relación del bucle más interno.

2.5.2.2 Reunión en bucle anidado por bloques (Block Nested-Loop Join).

Es una variante de la reunión en bucle anidado, donde se empareja cada bloque de la relación interna con cada bloque de la relación externa. En cada par de bloques se empareja cada tupla de un bloque con cada tupla del otro bloque para generar todos los pares de tuplas. De la misma forma, se añaden al resultado todas las parejas de tuplas que satisfacen la condición de reunión.

En el peor de los casos, cada bloque de la relación interna S se lee solamente una vez por cada bloque de la relación externa. El costo estimado de este enfoque es nBloques(R) + nBloques(R)*nBloques(S). Se observa que será más eficiente utilizar la relación más pequeña como la relación externa.

Otra mejora a esta estrategia es leer tanto bloques como sea posible de la relación más pequeña, digamos R, en el búfer de base de datos, guardando un bloque para la relación interna, y uno para la relación resultado. Si el búfer puede almacenar nBúfer bloques, entonces se deberían leer (nBúfer – 2) bloques desde R hacia el búfer cada vez y un bloque de S. El número total de R bloques accedidos es todavía nBloques(R), pero el número total de bloques S leídos se reduce a [nBloques(S)*(nBloques(R)/(nBúfer – 2))] aproximadamente. Con este enfoque el nuevo estimado de costo se convierte en:

nBloques(R) + [nBloques(S)*(nBloques(R)/(nBúfer – 2))]

En el mejor de los casos, cuando se pueden leer todos los bloques de R en el búfer, el costo se reduce a nBloques(R) + nBloques(S) [8]

2.5.2.3 Reunión en bucle anidado indexada (Indexed Nested-Loop Join).

Si se dispone de un índice o función de asociación sobre los atributos de reunión de la relación interna, se pueden sustituir las exploraciones ineficientes de archivo por búsquedas en el índice. Para cada tupla en R, se utiliza el índice para recuperar las tuplas de S que cumplan la condición de reunión.

En el peor de los casos sólo hay espacio en la memoria intermedia para una página de R y una página del índice. Por lo tanto, son necesarios nBloques(R) accesos a disco para leer la relación R. Para cada tupla de R, se realiza una búsqueda en el índice para S. Luego el costo de la reunión se calcula como nBloques(R) + nTuplas(R) * c, donde c es el costo de una única selección en S utilizando la condición de reunión. Esto muestra que el costo de recuperar las tuplas asociadas en S depende del tipo de índice y del número de tuplas correspondientes.

De la fórmula se observa que normalmente es más eficiente usar como relación más externa aquella que tenga menos tuplas.

2.5.2.4 Reunión por mezcla (Sort-Merge Join).

Este algoritmo se puede utilizar para calcular reuniones naturales y equirreuniones. Para las equirreuniones, la reunión más eficiente se consigue cuando ambas relaciones están ordenadas en los atributos de reunión. En este caso, se puede buscar por las tuplas que cumplan con los requisitos de R y S mezclando las dos relaciones. Si no están ordenadas, un procesamiento por adelantado se puede llevar a cabo para ordenarlas. Ya que las relaciones están ordenadas, se garantiza que las tuplas con el mismo valor de atributo de reunión están en orden consecutivo. Si se asume que la reunión es muchos a muchos, esto es, que puede haber muchas tuplas de R y S con el mismo valor de reunión, y si se asume que cada conjunto de tuplas con el mismo valor de reunión puede ser almacenado en el búfer de la base de datos al mismo tiempo, entonces cada bloque de cada relación necesita ser leído únicamente una vez.

Por lo tanto, el costo estimado de la reunión por mezcla es nBloques(R) + nBloques(S).

Si una relación tiene que ser ordenada, se tendría que añadir el costo de la ordenación, que se puede aproximar a nBloques(R)*[$\log_2$(nBloques(R))] [8]

2.5.2.5 Reunión por asociación (Hash Join).

Al igual que el algoritmo de reunión por mezcla, el algoritmo de reunión por asociación se puede utilizar para implementar reuniones naturales y

equirreuniones. El algoritmo utiliza una función de asociación *h()* para dividir las tuplas de ambas relaciones. La idea fundamental es dividir las tuplas de cada relación en conjuntos con el mismo valor de la función de asociación en los atributos de la reunión. Cada partición equivalente de R y S debería contener el mismo valor para los atributos de reunión, aunque pueden contener más de un valor. Por lo tanto, el algoritmo tiene que comprobar particiones equivalentes para el mismo valor.

Por ejemplo, si la relación R es dividida en R_1, R_2, .., R_M, y la relación S en S_1, S_2,...,S_M utilizando una función de asociación *h()*, entonces si B y C son atributos de R y S respectivamente, y h(R.B) ≠ h(S.C), entonces R.B ≠ S.C. Sin embargo, si h(R.B) = h(S.C), no implica necesariamente que R.B = S.C ya que valores diferentes pueden mapear al mismo valor de asociación.

La segunda fase, llamada la etapa de prueba, lee cada partición de R en turno y por cada una intenta reunir las tuplas en la partición con las tuplas en la partición S equivalente. Si una reunión por bucle anidado es utilizada para la segunda fase, la partición más pequeña es utilizada como el bloque externo, digamos R_i. La partición completa R_i es leída en memoria y cada bloque de la partición equivalente S_i es leído y cada tupla es usada para probar R_i por tuplas coincidentes.

El costo de la reunión por asociación se estima como 3(nBloques(R)+nBloques(S)). Esto representa tener que leer R y S para dividirlas, escribir cada partición a disco, y entonces tener que leer cada partición de R y S nuevamente para encontrar tuplas coincidentes.

Reunión en bucle anidado (Nested Loop).	nBloques(R) + nTuplas(R)*nBloques(S), si la memoria intermedia tiene únicamente un bloque para R y S. nBloques(R) + nBloques(S), si todos los bloques de R y S pueden ser leídos en la memoria intermedia.
Reunión en bucle anidado por bloques (Block Nested-Loop Join).	nBloques(R) + nBloques(R)*nBloques(S), si la memoria intermedia tiene únicamente un bloque para R y S. nBloques(R) + [nBloques(S)*(nBloques(R)/(nBúfer – 2))], si (nBúfer – 2) bloques para R. nBloques(R) + nBloques(S), si todos los bloques de R pueden ser leídos en la memoria intermedia.
Reunión en bucle anidado indexada (Indexed Nested-Loop Join).	nBloques(R) + nTuplas(R) * c, donde c es el costo de una única selección en S utilizando la condición de reunión.

Reunión por mezcla (Sort-Merge Join).	nBloques(R)*[$\log_2$(nBloques(R))] + nBloques(S)*[$\log_2$(nBloques(S))], para ordenación. nBloques(R) + nBloques(S), para mezcla.
Reunión por asociación (Hash Join).	3(nBloques(R)+nBloques(S)), si el índice de asociación es mantenido en memoria.

Tabla 2.5 Sumario del costo estimado de E/S de las estrategias para la operación Reunión.

2.6 Operación proyección. $\left(S = \prod_{A_1, A_2, \ldots, A_m}(R)\right)$

La operación proyección es una operación unaria que define una relación S conteniendo un subconjunto vertical de la relación R extrayendo los valores de los atributos especificados y eliminando los duplicados. [8]

Para implementar la proyección se requieren los siguientes pasos:

1) Eliminación de atributos no requeridos.
2) Eliminación de cualquiera tuplas duplicadas que son producidas del paso anterior. Este paso es requerido únicamente si los atributos de la proyección no incluyen una llave de la relación. Existen dos enfoques principales para eliminar duplicados; la ordenación y la asociación.

2.6.1 Cardinalidad de la proyección.

Cuando la proyección contiene un atributo clave, entonces ya que la eliminación de duplicados no es requerida, la cardinalidad de la proyección es nTuplas(S) = nTuplas(R).

Si la proyección consiste de un solo atributo no clave, se puede estimar la cardinalidad de la proyección como nTuplas (S) = $CS_A(R)$.

De otra forma si se asume que la relación es el producto Cartesiano de los valores de sus atributos, lo cual es generalmente poco realista, se podría estimar la cardinalidad como:

$$\text{nTuplas}(S) \leq \min\left(\text{nTuplas}(R), \prod_{i=1}^{m} \text{nDistinto}_{A_i}(R)\right)$$

2.6.1.1 Eliminación de duplicados mediante ordenación.

El objetivo de este acercamiento es ordenar las tuplas de la relación reducida utilizando todos los atributos restantes como la llave de ordenación. Esto

tiene el efecto de organizar las tuplas de tal forma que las duplicadas son adyacentes y pueden ser removidas fácilmente después de eso. Para remover los atributos no deseados, se requiere leer todas las tuplas de R y copiar los atributos requeridos a una relación temporal, a un costo de nBloques(R). El costo estimado de ordenación es nBloques(R) * [$\log_2$(nBloques(R))]; por lo que el costo combinado es nBloques(R) + nBloques(R) * [$\log_2$(nBloques(R))].

2.6.1.2 Eliminación de duplicados mediante asociación.

El objetivo de este enfoque puede ser útil si se tiene un gran número de bloques de búfer relativos al número de bloques de R. La asociación tiene dos etapas; división y eliminación de duplicados.

División.- Se asigna un bloque de búfer para leer la relación R, y (nBúfer - 1) bloques de búfer para la salida. Para cada tupla en R, se eliminan los atributos no deseados y entonces se aplica una función de asociación *h()* a la combinación de los atributos restantes, y se escribe la tupla reducida al valor asociado. La función de asociación *h()* debería ser escogida de forma que las tuplas sean uniformemente distribuidas a una de las (nBúfer - 1) particiones. Se garantiza que dos tuplas que pertenecen a diferentes particiones no son duplicados, porque tienen diferentes valores de asociación, lo cual reduce el área de búsqueda para la eliminación de duplicados a particiones individuales.

Eliminación de duplicados.- Procede como sigue:

Leer cada (nBúfer - 1) partición en turno.

Aplicar una segunda función de asociación $h_2()$ a cada tupla a medida que es leída.

Insertar el valor de asociación calculado en una tabla de asociación en memoria.

Si la tupla asocia al mismo valor de asociación como alguna otra tupla, comprobar si las dos son la mismas y eliminar la nueva si es un duplicado.

Una vez que una partición ha sido procesada, escribir las tuplas en la tabla de asociación al archivo resultado.

Si el número de bloques que se requieren para la tabla temporal que resulta de la proyección en R antes de la eliminación de duplicados es nb, el costo estimado es entonces nBloques(R) + nb.

Esto excluye escribir la relación resultado y asume que la asociación no requiere particiones de desbordamiento.

2.7 Evaluación de expresiones.

Para evaluar una expresión que contiene varias operaciones se consideran dos enfoques: [4]

- Materialización.- Si se aplica el enfoque de la materialización, se comienza por las operaciones de la expresión de nivel más bajo en un árbol de operadores. Se ejecutan estas operaciones y los resultados de cada operación intermedia se materializan; se almacenan en relaciones temporales para utilizarse en la evaluación de las operaciones del siguiente nivel. Para calcular el costo de evaluar una expresión con este enfoque, hay que añadir los costos de todas las operaciones, incluyendo el costo de escribir los resultados intermedios en disco.
- Encauzamiento.- Combina varias operaciones relacionales en un encauzamiento de operaciones, en el que se pasan los resultados de una operación a la siguiente operación del encauzamiento; eliminando el costo de leer y escribir relaciones temporales.

  Cada operación del encauzamiento toma un flujo de tuplas de sus entradas encauzadas y produce un flujo de tuplas como su salida. Para cada pareja de operaciones adyacentes en el encauzamiento se crea una memoria intermedia para guardar las tuplas que se envían de una operación a la siguiente. Sin embargo, como resultado del encauzamiento, las entradas de las operaciones no están disponibles todas a la vez para su procesamiento. Esto puede restringir la elección de algoritmos a emplear. [8]

2.8 Optimización de consultas.

La optimización de consultas es el proceso de selección del plan de evaluación de las consultas más eficiente de entre las muchas estrategias generalmente disponibles para el procesamiento de una consulta determinada.

Un aspecto de la optimización de las consultas tiene lugar en el nivel del álgebra relacional, donde el sistema intenta hallar una expresión que sea equivalente a la expresión dada, pero de ejecución más eficiente. Dada una expresión del álgebra relacional, es labor del optimizador de consultas diseñar un plan de evaluación de consultas que calcule el mismo resultado que la expresión dada minimizando el costo de la evaluación. [9]

La generación de planes de evaluación de consultas alternativos implica:

- Generación de expresiones lógicamente equivalentes a través de reglas de equivalencia.
- Anotación de expresiones resultantes para obtener planes de ejecución alternativos.

La generación de expresiones sólo es una parte del proceso de optimización de consultas, ya que cada operación de la expresión puede implementarse con algoritmos diferentes. Por tanto, se necesita un plan de evaluación que defina el algoritmo a utilizar para cada operación y la forma en que se ejecutarán las operaciones.

Además de considerar las expresiones alternativas de cada consulta, también hay que considerar los algoritmos alternativos para cada operación de cada expresión. Para escoger el mejor algoritmo global hay que considerar incluso los algoritmos no óptimos para operaciones individuales. Se pueden utilizar reglas muy parecidas a las reglas de equivalencia del álgebra relacional para definir los algoritmos que pueden utilizarse para cada operación, y si su resultado puede encauzarse o se debe materializar. Se pueden utilizar estas reglas para generar todos los planes de evaluación de consultas para una expresión dada.

Lo anterior sigue dejando el problema de la selección del mejor plan de evaluación de la consulta. Hay dos enfoques generales; el primero busca todos los

planes y escoge el mejor de una manera basada en costos. El segundo utiliza la heurística para escoger el plan. [9]

2.8.1 Optimización basada en costos.

Los optimizadores basados en costos generan una gama de planes de evaluación a partir de la consulta dada empleando las reglas de equivalencia y escogen la de costo mínimo.

Para consultas complejas el número de planes de consulta diferentes que son equivalentes a un plan dado puede ser grande. En general, con *n* relaciones hay *(2(n-1))!/(n-1)!* órdenes de reunión diferentes. [9]

Se puede desarrollar un algoritmo de programación dinámica para hallar los órdenes de reunión óptimos. Los algoritmos de programación dinámica almacenan los resultados de los cálculos y los reutilizan, un procedimiento que puede reducir enormemente el tiempo de ejecución.

El procedimiento de programación dinámica almacena los planes de evaluación que calcula en el arreglo asociado *mejorplan*, que está indexado por conjuntos de relaciones. Cada elemento del arreglo asociativo contiene dos componentes: el costo del mejor plan de *S* y el propio plan.

El procedimiento comprueba en primer lugar si el mejor plan para calcular la reunión del conjunto de relaciones dado *S* se ha calculado ya;

Si es así, devuelve el plan ya calculado.

Si no, el procedimiento intenta todas las maneras posibles de dividir *S* en dos subconjuntos disjuntos. Para cada división, el procedimiento halla de manera recursiva los mejores planes para cada uno de los dos subconjuntos y luego calcula el costo del plan global utilizando esa división.

El procedimiento escoge el plan más económico de entre todas las alternativas para dividir *S* en dos conjuntos.

Se ha demostrado que la complejidad temporal del procedimiento es de $O(3^n)$.

El orden en que la reunión de un conjunto de relaciones genera las tuplas también es importante para hallar el mejor orden global de reunión, ya que puede

afectar al costo de las reuniones posteriores. Se dice que un orden determinado de las tuplas es un orden interesante si puede resultar útil para alguna operación posterior.

Por lo tanto, no basta con hallar el mejor orden de reunión para cada subconjunto del conjunto de *n* relaciones dadas. Hay que hallar el mejor orden de reunión para cada subconjunto, para cada orden interesante de la reunión resultante para ese subconjunto.

El número de subconjuntos de *n* relaciones es 2^n. El número de órdenes de colocación interesantes no suele ser grande. Así, hay que almacenar alrededor de 2^n expresiones de reunión, es decir, la complejidad de espacio es del $O(2^n)$. El algoritmo de programación dinámica para hallar el mejor orden de reunión puede extenderse de manera sencilla para que trabaje con los órdenes de colocación. El costo del algoritmo extendido depende del número de órdenes interesantes para cada subconjunto de relaciones; dado que se ha hallado que este número en la práctica es pequeño el costo se queda en $O(3^n)$.

2.8.2 Optimización heurística.

Muchos sistemas utilizan la heurística para reducir el número de elecciones que hay que hacer de una manera basada en costos. Algunos sistemas deciden incluso utilizar sólo la heurística y no utilizan en absoluto la optimización basada en el costo.

La optimización heurística o basada en reglas, transforma un árbol de consulta dado mediante un conjunto de reglas que típicamente, aunque no en todos los casos, mejoran el desempeño de la ejecución.

La regla para la transformación de consultas del álgebra relacional “llevar a cabo las operaciones de selección tan pronto como sea posible” es un ejemplo de regla heurística.

Etapas de un algoritmo típico de optimización heurística. [9]

1) Descomponer las selecciones conjuntivas en una secuencia de operaciones de selección sencillas. Este paso, facilita el desplazamiento de las operaciones de selección hacia la parte inferior del árbol de consultas.

2) Hay que sustituir por operaciones de reunión las operaciones producto cartesiano seguidas de condiciones de selección.
3) Hay que determinar las operaciones de selección y de reunión que producirán las relaciones de menor tamaño, es decir, que producirán las relaciones con el menor número de tuplas, para conseguir su ejecución lo antes posible.
4) Hay que dividir las listas de atributos de proyección y desplazarlas hacia la parte inferior del árbol todo lo que sea posible, creando proyecciones nuevas cuando sea necesario.
5) Calcular las expresiones comunes una sola vez. Si una expresión común aparece más de una vez en el árbol, y el resultado que produce no es demasiado grande, almacenar el resultado después de que ha sido calculado y reutilizarlo cuando sea requerido puede resultar benéfico.

Las heurísticas citadas reordenan la representación inicial del árbol de consultas de manera que las operaciones que reducen el tamaño de los resultados intermedios se apliquen antes; las selecciones tempranas reducen el número de tuplas y las proyecciones tempranas reducen el número de atributos. Las transformaciones heurísticas también reestructuran el árbol de modo que el sistema lleve a cabo las operaciones de selección y de reunión más restrictivas antes que otras operaciones parecidas.

La optimización heurística hace corresponder aún más la expresión de consulta transformada heurísticamente con secuencias alternativas de operaciones para producir un conjunto candidato de planes de evaluación. Cada plan incluye: las operaciones relacionales que hay que llevar a cabo, los índices que hay que emplear, el orden en el que hay que tener acceso a las tuplas y el orden en el que hay que realizar las operaciones.

La fase de selección del plan de acceso del optimizador heurístico selecciona la estrategia más eficiente para cada operación.

2.9 Administración del desempeño en Oracle.

La afinación comienza en la planeación del sistema y fases de diseño y continúa a lo largo de la vida del sistema.

Los objetivos de afinación varían dependiendo de las necesidades de la aplicación. Las aplicaciones On-Line Transaction Processing OLTP definen el desempeño en términos del procesamiento. El procesamiento del sistema es igual a la cantidad de trabajo llevada a cabo en un tiempo determinado. Estas aplicaciones deben procesar miles o incluso millones de transacciones muy pequeñas por día.

Las aplicaciones Decision Support Systems DSS definen el desempeño en términos del tiempo de respuesta. El tiempo de respuesta es igual al tiempo de servicio más el tiempo de espera. Las demandas en la base de datos que son realizadas por usuarios de aplicaciones DSS varían dramáticamente. En un momento pueden ingresar una consulta que trae únicamente unos pocos registros y el momento siguiente pueden ingresar una consulta paralela masiva que trae y ordena cientos de miles de registros de diferentes tablas. [10]

2.9.1 Problemas de desempeño.

El desempeño depende de cuántos recursos están disponibles, cuántos clientes requieren el recurso, cuánto deben esperar por el recurso y qué tanto tiempo poseen el recurso.

Los problemas de desempeño ocurren cuando una tarea toma más para realizarse que el tiempo permitido. El problema ocurre porque un recurso de un tipo particular es insuficiente o inadecuado. El recurso puede ser un recurso físico, como los búferes de memoria disponibles que almacenan bloques de datos; o artificial, como un bloqueo.

Los problemas comunes de desempeño son los siguientes: [10]

1. El tiempo de espera se incrementa a medida que la contención aumenta. Contención es la competencia por recursos.
2. A medida que el número de unidades solicitadas aumenta, el tiempo para completar el servicio se incrementa.

3. Efectos de la demanda excesiva:
    a. Tiempo de respuesta ampliamente incrementado.
    b. Procesamiento reducido.

2.9.2 Escalabilidad.

Es la habilidad del sistema para procesar más carga de trabajo con un incremento proporcional en el uso de los recursos del sistema. En otras palabras, si se duplica la carga de trabajo en un sistema escalable, entonces el sistema debe utilizar dos veces tantos recursos del sistema. Si una aplicación agota un recurso del sistema al grado que el procesamiento adicional es imposible cuando se incrementa la carga de trabajo del sistema, se dice que la aplicación no es escalable. Esto puede resultar en procesamientos fijos y tiempos de respuesta pobres. [11]

2.9.3 Metodología de afinación.

La tabla siguiente muestra las fases del ciclo de desarrollo que pueden ser afinadas así como el responsable de llevarlo a cabo. Los pasos están organizados en orden de rendimiento descendente, es decir, los pasos con el mayor efecto en el desempeño aparecen primero. [12]

Analista de negocio.	1. Afinar la función del negocio.
Diseñador.	2. Afinar el diseño de los datos. 3. Afinar el diseño de proceso.
Desarrollador de aplicaciones.	4. Afinar sentencias SQL. 1) Identificar las sentencias SQL problemáticas. 2) Verificar las estadísticas del optimizador. 3) Revisar los planes de ejecución. 4) Reestructurar las sentencias SQL. 5) Reestructurar los índices. 6) Mantener los planes de ejecución a través del tiempo. 5. Afinar la estructura física.
Administrador de la base de datos.	6. Afinar la asignación de memoria. 7. Afinar la E/S. 8. Afinar la contención de memoria.
Administrador del sistema operativo.	9. Afinar el sistema operativo.
Administrador de redes.	10. Problemas de red.

Tabla 2.6 Roles de afinación.

2.10 Procesamiento de consultas SQL.

2.10.1 System Global Area (SGA).

Es un área de memoria utilizada para almacenar información de la base de datos que es compartida por los procesos de base de datos. Es asignada en la memoria virtual de la computadora donde reside el servidor de Oracle. El SGA contiene varias sub-áreas que son usadas para organizar y mantener diferentes tipos de información. [13]

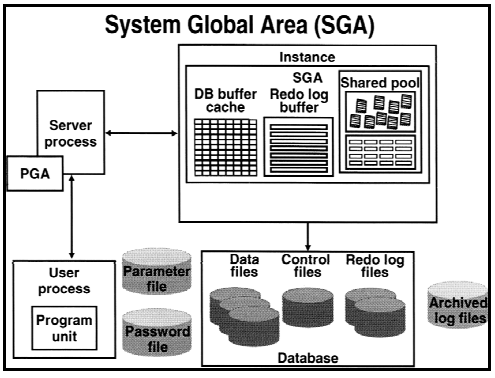

Figura 2.2 System Global Area.

Las principales estructuras de datos del SGA son:

- Database buffer cache, estructura de memoria en el SGA que almacena los bloques de datos más recientemente utilizados.
- Redo log buffer, estructura de memoria en el SGA que almacena una bitácora de cambios hechos a la base de datos.
- Shared pool, porción del SGA que contiene estructuras de memoria como áreas SQL compartidas.

Los componentes principales del shared pool son la librería caché y el caché del diccionario de datos. La librería caché almacena la forma ejecutable del recientemente referenciado SQL y código PL/SQL. El caché del diccionario de datos almacena información sobre los objetos en la base de datos, la estructura de la base de datos y las definiciones de columna, así como los usuarios válidos de la base de datos y sus privilegios.

La administración automática de memoria compartida en Oracle 10g simplifica la configuración del SGA y es la configuración de memoria recomendada. Para utilizarla, se establece el parámetro de inicio SGA_TARGET a un valor que no equivale a cero y también se establece el parámetro de inicio STATISTICS_LEVEL a TYPICAL u ALL. El valor del parámetro SGA_TARGET debería ser establecido a la cantidad de memoria que se quiere dedicar para el SGA. En respuesta a la carga de trabajo en el sistema, la administración automática del SGA distribuye la memoria adecuadamente para los pools siguientes:

- Database buffer cache (DB_CACHE_SIZE).
- Shared pool (SHARED_POOL_SIZE).
- Large pool (LARGE_POOL_SIZE).
- Java pool (JAVA_POOL_SIZE).
- Streams pool (STREAMS_POOL_SIZE).

El parámetro SGA_TARGET es dinámico y puede ser modificado utilizando el comando ALTER SYSTEM. [14]

2.10.2 Áreas SQL compartidas.

Cuando se invoca código de aplicación, la base de datos Oracle intenta reusar código existente si ha sido ejecutado previamente de tal manera que pueda ser compartido. Si la representación analizada de la sentencia existe en la librería caché y si puede ser compartida, la base de datos Oracle reutiliza el ejecutable existente. [15]

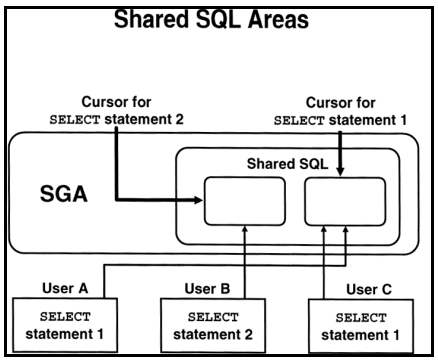

Figura 2.3 Áreas SQL compartidas.

Dentro del área SQL compartida, cada sentencia SQL es analizada en su propia parte, conocida como área de contexto o cursor.

Cada cursor mantiene la siguiente información:

a) La sentencia analizada.
b) El plan de ejecución.
c) Una lista de objetos referenciados.

Si dos usuarios emiten la misma sentencia SQL, entonces pueden usar el mismo cursor.

2.10.3 Componentes principales para el procesamiento de una sentencia SQL.

El analizador verifica el análisis sintáctico y semántico.

El optimizador utiliza métodos de estimación de costos, optimizador basado en costos (CBO), o reglas internas, optimizador basado en reglas (RBO), para determinar la forma más eficiente de producir el resultado de la consulta.

El generador fuente de registros recibe el plan óptimo del optimizador y produce el plan de ejecución para la sentencia SQL.

El motor de ejecución SQL opera en el plan de ejecución asociado con una sentencia SQL y produce los resultados de una consulta. [16]

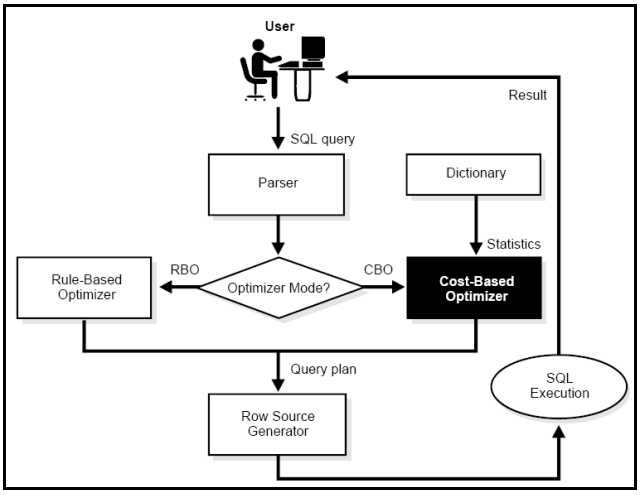

Figura 2.4 Esquema general de procesamiento SQL.

2.10.4 Fases de procesamiento de sentencias SQL.

1. Análisis.

Cuando una aplicación emite una sentencia SQL, la aplicación hace una llamada de análisis a la base de datos Oracle. Durante esta llamada la base de datos Oracle:

- Verifica la sentencia por validez sintáctica y semántica.
- Determina si el proceso que emite la sentencia tiene privilegios para ejecutarlo.
- Asigna un área SQL privada para la sentencia. Oracle representa cada sentencia que ejecuta con un área SQL compartida y un área SQL privada. Oracle reconoce cuando dos usuarios están ejecutando la misma sentencia SQL y reutiliza el área SQL compartida para esos usuarios. Sin embargo, cada usuario debe tener una copia separada del área SQL privada de la sentencia. Por lo tanto, varias áreas SQL privadas pueden ser asociadas con la misma área SQL compartida. [17]

- Determina si existe un área SQL compartida que contenga la representación analizada de la sentencia en la librería caché.
  a. Si es así, el proceso de usuario utiliza esta representación analizada y ejecuta la sentencia inmediatamente. Cuando una sentencia SQL se encuentra en un área SQL compartida, entonces la fase de análisis se abrevia y el cursor existente es utilizado. Este proceso se conoce como "soft parse".
  b. Si no, la base de datos Oracle genera la representación analizada de la sentencia y el proceso de usuario asigna un área compartida SQL para la sentencia en la librería caché. Este proceso se conoce como "hard parse". Después de que un área SQL compartida ha sido asignada para una sentencia, puede ser ejecutada repetidamente sin ser reanalizada.

2. Vinculación.

- EL servidor Oracle examina la sentencia por referencias a variables de vinculación.
- El servidor Oracle asigna o reasigna un valor a cada variable.

3. Ejecución.

- El servidor Oracle aplica el árbol de análisis a los búferes de datos.
- Múltiples usuarios pueden compartir el mismo árbol de análisis.
- El servidor Oracle realiza lecturas físicas o lógicas, leyendo o escribiendo para sentencias DML y también ordenando los datos cuando es necesario.

4. Recuperación.

Durante esta fase el servidor Oracle recupera los registros para una sentencia SELECT. Esta fase aplica únicamente a consultas. [18]

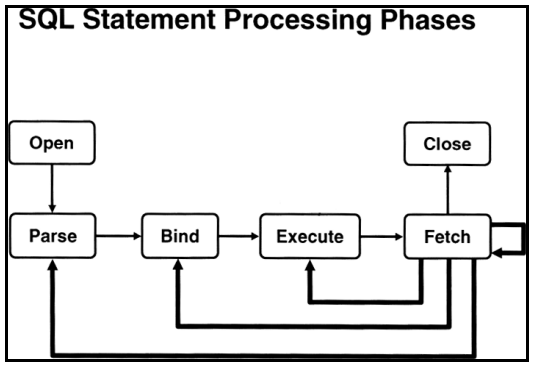

Figura 2.5 Fases de procesamiento de una sentencia SQL.

2.11 SQL*Plus AUTOTRACE.

En SQL*Plus se puede obtener automáticamente el plan de ejecución y algunas estadísticas adicionales en la corrida de un comando SQL utilizando la opción de AUTOTRACE. A diferencia del comando EXPLAIN PLAN, la sentencia es realmente ejecutada. Sin embargo se puede elegir suprimir la ejecución de la sentencia especificando AUTROTRACE TRACEONLY EXPLAIN.

AUTOTRACE es una herramienta de diagnóstico excelente para la afinación de sentencias SQL. [19]

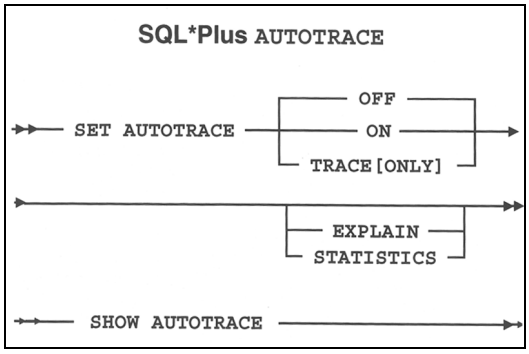

Figura 2.6 Sintaxis de SQL*Plus AUTOTRACE.

OFF	Deshabilita el auto rastreo de sentencias SQL.
ON	Habilita el auto rastreo de sentencias SQL.
TRACEONLY	Habilita el auto rastreo y suprime la salida de la sentencia.
EXPLAIN	Despliega el plan de ejecución pero no despliega estadísticas.
STATISTICS	Despliega estadísticas pero no despliega el plan de ejecución.

Tabla 2.7 Opciones del comando AUTOTRACE.

Si ambas opciones EXPLAIN y STATISTICS son omitidas, los planes de ejecución y estadísticas son desplegados por defecto.

2.11.1 Prerrequisitos para utilizar AUTOTRACE.

Para utilizar la opción EXPLAIN de AUTOTRACE, se debe crear primero la tabla del plan en el esquema, utilizando el script utlxplan.sql.

Para acceder la información de STATISTICS, se debe tener acceso a varias tablas de desempeño dinámico. El DBA puede otorgar el acceso utilizando el rol plustrace creado en el script plustrce.sql.

El plan de ejecución generado por la opción EXPLAIN de la herramienta AUTOTRACE muestra la secuencia de operaciones determinada por el optimizador de Oracle para ejecutar la sentencia.

2.11.2 Leyendo los planes de ejecución.

El árbol fuente de registros es el núcleo del plan de ejecución, el cual muestra los siguientes elementos representados en la Figura 2.7 Principales elementos del reporte generado por la herramienta AUTOTRACE.:

1. Un ordenamiento de las tablas referenciadas por la sentencia.
2. Un método de acceso por cada tabla mencionada en la sentencia.
3. Un método de reunión para tablas afectadas por operaciones de reunión en la sentencia.
4. Operaciones de datos como filtro, ordenación o agregación.
5. Información como el costo y la cardinalidad de cada operación.
6. Estadísticas de base de datos que muestran los recursos del sistema requeridos para ejecutar la sentencia. Estas estadísticas se describen en la Tabla 2.8 Estadísticas de SQL*Plus AUTOTRACE. [

El orden de ejecución en la salida de EXPLAIN PLAN comienza con la línea que es la más lejana identada a la derecha. El siguiente paso es el padre de esa línea. Si dos líneas están igualmente identadas, entonces la línea superior es ejecutada primero normalmente.

Cada paso del plan de ejecución devuelve un conjunto de registros que ya sea es utilizado por el siguiente paso o, en el último paso, es devuelto al usuario o aplicación emitiendo la sentencia SQL.

La numeración de los pasos refleja el orden en el cual son desplegados en respuesta a la opción EXPLAIN. [20]

```
Connected to:
Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - 64bit Production
With the Partitioning, OLAP and Data Mining options

SQL> set linesize 180
SQL> set autotrace traceonly
SQL> --GEXP_EDO_GET_DOCDATAATTRB_VAL
SQL> SELECT DISTINCT /* TAG 6 */ DD.ATTRIBUTE_VALUE
  2             FROM GEXT_DOCUMENT_DETAIL DD, GEXT_DOCUMENT_MASTER DM
  3            WHERE (    (DD.DOCUMENT_ID = DM.DOCUMENT_ID)
  4                   AND (DD.DOCUMENT_VERSION_ID = DM.DOCUMENT_VERSION_ID)
  5                   AND (DM.DOCUMENT_ID = 'GR0125')
  6                   AND (DD.ATTRIBUTE_ID = 1516));


Execution Plan
----------------------------------------------------------
Plan hash value: 2035648940

                                                                                                    5
-------------------------------------------------------------------------------------------------------------
| Id  | Operation                     | Name                       | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT              |                            |     1 |    37 |     5  (20)| 00:00:01 |
|   1 |  HASH UNIQUE                  |                            |     1 |    37 |     5  (20)| 00:00:01 |
|   2 |   NESTED LOOPS                |                            |     1 |    37 |     4   (0)| 00:00:01 |
|*  3 |    INDEX RANGE SCAN           | GEXI_DOCUMENT_MASTER_PK    |     1 |    13 |     2   (0)| 00:00:01
|   4 |    TABLE ACCESS BY INDEX ROWID| GEXT_DOCUMENT_DETAIL       |     1 |    24 |     2   (0)| 00:00:0
|*  5 |     INDEX UNIQUE SCAN         | GEXI_DOCUMENT_DETAIL1_PK   |     1 |       |     1   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------------------

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

   3 - access("DM"."DOCUMENT_ID"='GR0125')
   5 - access("DD"."DOCUMENT_ID"='GR0125' AND "DD"."DOCUMENT_VERSION_ID"="DM"."DOCUMENT_VERSION_ID
              " AND "DD"."ATTRIBUTE_ID"=1516)


Statistics
----------------------------------------------------------
       1453  recursive calls
          0  db block gets
        435  consistent gets
         16  physical reads
          0  redo size
        232  bytes sent via SQL*Net to client
        239  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
         49  sorts (memory)
          0  sorts (disk)
          1  rows processed
```


Figura 2.7 Principales elementos del reporte generado por la herramienta AUTOTRACE.

Recursive Calls	Número de llamadas recursivas generadas en los niveles de usuario y de sistema. Las llamadas recursivas son sentencias SQL que se llevan a cabo en nombre de la sentencia SQL en cuestión. La base de datos Oracle mantiene tablas que se utilizan para procesamiento interno. Cuando la base de datos Oracle requiere hacer un cambio a estas tablas, genera una sentencia SQL interna, que a su vez genera una llamada recursiva. Por ejemplo, si se tuviera que analizar sintácticamente la consulta, se tendrían que ejecutar otras consultas para obtener información del diccionario de datos, éstas serían llamadas recursivas.
Db Block Gets*	Esta estadística representa el número de veces que un bloque actual fue solicitado. Los bloques en modo actual son recuperados como existen al momento. Por ejemplo, durante una consulta, se pueden observar lecturas en modo actual debido a la lectura del diccionario de datos para encontrar la información de una tabla para realizar una lectura completa, porque se necesita la información del momento, no la lectura consistente. Durante una modificación, se accederán los bloques en modo actual para escribirlos. La suma de esta estadística junto con Consistent Gets se reporta como lecturas lógicas.
Consistent Gets*	Número de veces que una lectura consistente fue solicitada para un bloque. Usualmente, los bloques son recuperados en modo consistente para consultas. Incluye la cuenta de bloques leídos de los segmentos de rollback para deshacer un bloque.
Physical Reads*	Número de bloques leídos desde disco.
Redo Size	Cantidad de redo generado en bytes para sentencias DML.
Bytes Sent Via SQL*Net To Client	Tamaño colectivo en bytes del conjunto resultado.
Bytes Received Via SQL*Net From Client	Tamaño de la consulta según transmitida en la red.
SQL*Net Roundtrips To/From Client	Número total de interacciones que se llevaron a cabo entre el cliente y el servidor para obtener la respuesta.
Sorts (Memory)	Número de ordenaciones ejecutadas en memoria.
Sorts (Disk)	Número de ordenaciones ejecutadas utilizando almacenamiento de disco temporal.
Rows Processed	Número total de registros devueltos por la sentencia SELECT o modificados por la sentencia INSERT, UPDATE, o DELETE.
* Son las 3 estadísticas que son usualmente seguidas de cerca. Las ordenaciones deberían ser ejecutadas en memoria en lugar de en disco.	

Tabla 2.8 Estadísticas de SQL*Plus AUTOTRACE. [21]

2.12 Introducción al optimizador.

El optimizador determina la manera más eficiente de ejecutar una sentencia SQL considerando varios factores relacionados a los objetos referenciados y las condiciones especificadas en la consulta. La salida del optimizador es un plan que describe un método óptimo de ejecución.

Existen frecuentemente diferentes formas de ejecutar una sentencia SQL; el procedimiento que Oracle emplea para ejecutar una sentencia puede afectar ampliamente que tan rápido se ejecuta la sentencia.

El optimizador considera muchos factores entre las rutas de acceso alternativas. Puede utilizar un enfoque basado en costos (CBO) o basado en reglas (RBO). Se pueden influenciar las elecciones del optimizador estableciendo el objetivo y el enfoque del optimizador y reuniendo estadísticas representativas para el CBO. [22]

2.12.1 Plan de ejecución.

La combinación de los pasos que Oracle emplea para ejecutar una sentencia se llama plan de ejecución. Un plan de ejecución incluye una ruta de acceso para cada tabla que la sentencia accede y un ordenamiento de las tablas (orden de reunión) con el método de reunión apropiado. [22]

2.12.2 Optimizador basado en costos (CBO).

El CBO fue introducido en Oracle 7. [23] El CBO determina cuál es el plan de ejecución más eficiente, considerando rutas de acceso disponibles y teniendo en cuenta información basada en estadísticas para los objetos del esquema accedidos por la sentencia. El CBO también considera hints, que son sugerencias de optimización colocadas en un comentario dentro de la sentencia.

El CBO ejecuta los pasos siguientes:

1. El optimizador genera un conjunto de planes potenciales para la sentencia SQL basado en rutas de acceso disponibles y hints.
2. El optimizador estima el costo de cada plan basado en estadísticas en el diccionario de datos para la distribución de datos y características de almacenamiento de las tablas, índices, y particiones accedidos por la sentencia.

El costo utilizado por el optimizador de consultas representa un estimado del número de E/S de disco y la cantidad de CPU y memoria empleada en realizar una operación.

3. El optimizador compara los costos de los planes y elige aquél con el costo menor. [24]

Para mantener la eficacia del CBO, se deben reunir estadísticas y mantenerlas actualizadas. [25]

2.12.2.1 Eligiendo un enfoque y objetivo para el optimizador.

Por defecto, el objetivo del CBO es el mejor procesamiento. Esto significa que elige la menor cantidad de recursos necesarios para procesar todos los registros accedidos por la sentencia.

Oracle puede también optimizar una sentencia con el objetivo de mejor tiempo de respuesta. Esto significa que utiliza la menor cantidad de recursos necesarios para procesar el primer registro accedido por una sentencia SQL.

Se debe elegir un objetivo para el optimizador basado en las necesidades de la aplicación.

El parámetro de inicio OPTIMIZER_MODE establece el comportamiento por defecto para elegir un enfoque de optimización para la instancia de Oracle. [26]

CHOOSE (Oracle 7 a Oracle 9i)
El optimizador elige entre un enfoque basado en costos y un enfoque basado en reglas dependiendo si están disponibles estadísticas. Si el diccionario de datos contiene estadísticas para al menos una de las tablas accedidas, entonces el optimizador utiliza un enfoque basado en costos y optimiza con el objetivo de mejor procesamiento, de lo contrario el optimizador utiliza el enfoque basado en reglas.
ALL_ROWS (Oracle 7 a Oracle 10g)
El optimizador utiliza un enfoque basado en costos para todas las sentencias SQL en la sesión independientemente de la presencia de estadísticas y optimiza con el objetivo de mejor procesamiento.
RULE (Oracle 7 a Oracle 10g)
El optimizador elige un enfoque basado en reglas para todas las sentencias SQL independientemente de la presencia de estadísticas.

Tabla 2.9 Valores del parámetro de inicio OPTIMIZER_MODE.

Si el optimizador utiliza el enfoque basado en costos para la sentencia SQL, y si algunas tablas accedidas por la sentencia no tienen estadísticas, entonces el optimizador utiliza información interna, como el número de bloques de datos asignados a esas tablas, para estimar otras estadísticas para esas tablas.

2.12.2.2 Rutas de acceso para el CBO.

Una de las más importantes elecciones que hace el optimizador cuando formula un plan de ejecución es cómo recuperar datos desde la base de datos. Para cualquier registro en cualquier tabla accedida por una sentencia SQL, puede haber muchas rutas de acceso por las cuales el registro puede localizarse y recuperarse. El optimizador elige una de ellas. [27]

1. Lectura completa de tabla. (Full Table Scans)

Lee todos los registros de una tabla y filtra aquellos que no cumplen el criterio de selección. Cuando Oracle realiza una lectura completa de tabla, los bloques son leídos secuencialmente. Ya que los bloques son adyacentes, pueden usarse llamadas de E/S más grandes que un solo bloque para acelerar el proceso. El tamaño de las llamadas de lectura puede ir desde un bloque hasta el número de bloques definidos por el parámetro de inicio DB_FILE_MULTIBLOCK_READ_COUNT.

2. Lectura por ROWID. (ROWID Scans)

El ROWID de un registro especifica el archivo de datos y bloque de datos que contiene el registro así como la localización del registro en ese bloque. Localizar un registro especificando su ROWID, es la forma más rápida de recuperar un único registro, porque la localización exacta del registro en la base de datos es especificada.

Para acceder una tabla por ROWID, Oracle primero obtiene los ROWIDs de los registros seleccionados, ya sea de la cláusula WHERE de la sentencia o a través de una lectura de índice de uno o más de los índices de la tabla. Oracle entonces localiza cada registro seleccionado en la tabla basado en su ROWID.

3. Lectura de Índice. (Index Scans)

En este método, un registro se recupera recorriendo el índice, utilizando los valores de columna indexados especificados en la sentencia. Si la sentencia accede únicamente columnas del índice, Oracle lee los valores de columna indexados directamente del índice, en lugar de la tabla.

El índice contiene el valor indexado además de los ROWIDs de los registros en la tabla que tienen ese valor. Por lo tanto, si la sentencia accede a otras

columnas además de las columnas indexadas, el servidor Oracle puede encontrar los registros en la tabla utilizando una lectura por ROWID.

Lectura de índice único. Index Unique Scans.
Esta lectura devuelve un único ROWID y se realiza cuando la sentencia contiene una restricción de PRIMARY KEY o UNIQUE garantizando que un único registro es accedido. Esta ruta de acceso se utiliza cuando todas las columnas del índice único son especificadas con condiciones de igualdad.
Lectura de índice de rango. Index Range Scans.
Esta lectura es una operación común para acceder datos selectivos. Puede ser limitada en ambos lados o ilimitada en uno o ambos lados. Los datos son devueltos en orden ascendente de las columnas del índice. Múltiples registros con valores idénticos son ordenados en orden ascendente por ROWID. El optimizador utiliza una lectura de rango cuando encuentra una o más columnas delanteras de un índice especificadas en condiciones como col1 = :b1, col1 < :b1, col1 > :b1 y cualquier combinación de las condiciones anteriores para columnas delanteras en el índice.
Lectura de índice rápida y completa. Fast Full Index Scans.
Es una alternativa a la lectura de tabla completa cuando el índice contiene todas las columnas que son necesarias para la consulta y al menos una columna en la llave del índice tiene la restricción NOT NULL. Esta lectura accede a los datos en el índice sin acceder a la tabla. Esta lectura lee el índice completo utilizando lecturas multibloque y puede ser paralelizada.

Tabla 2.10 Tipos de lectura índice. [27]

2.12.3 Reuniones.

Las reuniones son sentencias que recuperan datos de más de una tabla. Una reunión se distingue por múltiples tablas en la cláusula FROM, y la relación entre las tablas se define a través de la existencia de una condición de reunión en la cláusula WHERE.

Para ejecutar una sentencia que reúne más de dos tablas, Oracle reúne dos de las tablas y entonces reúne la fuente de datos resultante con la siguiente tabla. Este proceso conocido como orden de reunión, continúa hasta que todas las tablas son reunidas al resultado. [28]

2.12.3.1 Tipos de operaciones de reunión.

Reunión en Bucle Anidado. Nested Loop Joins. [28]
El optimizador utiliza reuniones en bucle anidado cuando se reúne un número pequeño de registros, y la condición de reunión es una forma eficiente de acceder la segunda tabla. Una reunión en bucle anidado involucra los pasos siguientes: I. El optimizador determina la tabla que conduce y la designa como la tabla externa. II. La otra tabla es designada como la tabla interna. III. Por cada registro en la tabla externa, Oracle accede todos los registros en la tabla interna. El bucle externo es por cada registro en la tabla externa y el bucle interno es por cada registro en la tabla interna. El bucle externo aparece antes del bucle interno en el plan de ejecución.
Reunión por Asociación. Hash Joins.

Las reuniones por asociación se emplean para reunir grandes conjuntos de datos utilizando equirreuniones. El optimizador utiliza la más pequeña de dos tablas o fuentes de datos para construir una tabla de asociación de la llave de reunión en memoria. Entonces se lee la tabla más grande, probando la tabla de asociación para encontrar los registros reunidos.

Sin embargo, si la tabla de asociación crece demasiado para caber en memoria, el optimizador la descompone en diferentes particiones.

Reunión por Mezcla. Sort Merge Joins.

Las reuniones por mezcla son útiles cuando la condición de reunión entre dos tablas es una condición de desigualdad como <, <=, >, >=. Las reuniones por mezcla se desempeñan mejor que las reuniones en bucle anidado para grandes conjuntos de datos.

La reunión consiste de dos pasos:

I. Operación de reunión ordenación, ambas entradas se ordenan en la llave de reunión.
II. Operación de reunión mezcla, las listas ordenadas se mezclan.

El optimizador puede elegir una reunión por mezcla sobre una reunión por asociación para reunir grandes cantidades de datos si alguna de las siguientes condiciones es verdadera:

- La condición de reunión entre dos tablas no es una equirreunión.
- Debido a ordenaciones ya requeridas por otras operaciones, el optimizador encuentra más barato emplear una reunión por mezcla que una reunión por asociación.

Reunión Externa. Outer Joins.

Una reunión externa amplía el resultado de una simple reunión. Una reunión externa devuelve todos los registros que satisfacen la condición de reunión y también devuelve algunos o todos los registros de una tabla para los cuales ningún registro de la otra satisface la condición de reunión.

2.12.4 Estableciendo parámetros para el CBO.

Controlando el comportamiento del CBO. [29]

CURSOR_SHARING
Valor por defecto: EXACT (8i, 9i, 10g)

Este parámetro convierte valores literales en sentencias SQL a variables de vinculación. Convertir los valores mejora el compartir cursores y puede afectar los planes de ejecución de sentencias SQL. El optimizador genera el plan de ejecución basado en la presencia de variables de vinculación y no en los valores literales actuales.

DB_FILE_MULTIBLOCK_READ_COUNT
Valor por defecto: 8 (8i, 9i, 10g)

Este parámetro especifica el número de bloques que son leídos en una sola operación de E/S durante una lectura completa de tabla o una lectura de índice rápida y completa. Valores grandes de este parámetro resultan en un menor costo para lecturas completas de tabla y pueden resultar en el optimizador eligiendo una lectura completa de tabla sobre una lectura de índice. Si este parámetro no se establece explícitamente en Oracle 10g o se establece a cero, el optimizador utilizará el valor por defecto de 8.

OPTIMIZER_MODE
Valor por defecto: CHOOSE (8i, 9i), ALL_ROWS (10g)

Este parámetro de inicio establece el modo del optimizador al iniciar la instancia. Los valores posibles en Oracle 8i y 9i son RULE, CHOOSE, ALL_ROWS, FIRST_ROWS_n y FIRST_ROWS. Para Oracle 10g son ALL_ROWS, FIRST_ROWS_n y FIRST_ROWS.

PGA_AGGREGATE_TARGET
Valor por defecto: 10MB o 20% del tamaño del SGA, cualquiera que sea más grande (10g), 0 (9i)

Este parámetro controla automáticamente la cantidad de memoria asignada para ordenaciones y reuniones por asociación. Grandes cantidades de memoria asignadas para ordenaciones y reuniones por asociación reducen el costo del optimizador para estas operaciones. Para utilizar este parámetro, el parámetro WORK_AREA_SIZE debe ser establecido a AUTO.

WORK_AREA_SIZE
Valor por defecto: AUTO (10g), derivado de PGA_AGGREGATE_TARGET (9i)

Especifica la política para los tamaños de las áreas de trabajo. Si se establece a MANUAL, el único método disponible antes de 9i, las áreas de trabajo son controladas por los parámetros *_AREA_SIZE. Si se establece a AUTO, las áreas de trabajo son controladas por el parámetro PGA_AGGREGATE_TARGET.

2.12.5 Administración de memoria PGA.

El área global de programa (Program Global Area) es una región de memoria privada que contiene datos e información de control para un proceso servidor. El acceso a ella es exclusivo a ese proceso servidor y es leída y escrita únicamente por el código Oracle que actúa en su representación. Un ejemplo de tal información es el área de ejecución de un cursor. Cada vez que se ejecuta un cursor, una nueva área de ejecución se crea para ese cursor en la región de memoria PGA del proceso servidor ejecutando ese cursor.

Para consultas complejas, una gran porción del área de ejecución se dedica a áreas de trabajo asignadas por operadores intensivos de memoria, como los operadores basados en ordenación o los operadores de reuniones por asociación.

El tamaño de un área de trabajo puede ser controlado y afinado. Idealmente el tamaño de un área de trabajo es lo suficientemente grande para acomodar los datos de entrada y las estructuras de memoria auxiliares asignadas por su operador SQL asociado.

La administración automática de memoria PGA simplifica y mejora la forma en que se asigna la memoria PGA. Por defecto, la administración de memoria PGA está habilitada en Oracle 10g. El DBA necesita simplemente especificar el tamaño total dedicado a la memoria PGA para la instancia Oracle, estableciendo el parámetro de inicio PGA_AGGREGATE_TARGET.

Cuando se corre bajo el modo de administración de memoria PGA automática, dimensionar las áreas de trabajo para todas las sesiones se vuelve automático y los parámetros *_AREA_SIZE son ignorados por todas las sesiones ejecutándose en ese modo.

Por compatibilidad anterior, la administración automática de memoria PGA puede deshabilitarse estableciendo el valor del parámetro de inicio PGA_AGGREGATE_TARGET a cero. Cuando esto sucede, el tamaño máximo de

un área de trabajo puede dimensionarse con el parámetro *_AREA_SIZE asociado.

El modo de administración de memoria PGA automático aplica únicamente a áreas de trabajo asignadas por servidores Oracle dedicados. El tamaño de las áreas de trabajo asignadas por servidores Oracle compartidos es controlado todavía por los viejos parámetros *_AREA_SIZE ya que estas áreas de trabajo son asignadas principalmente en SGA y no en PGA. [30]

2.12.6 Optimización basada en reglas.

El RBO era el único optimizador disponible antes de Oracle 7. [31]

Utilizando la optimización basada en reglas, el optimizador elige un plan de ejecución basado en las rutas de acceso disponibles y las clasificaciones de esas rutas de acceso. La clasificación de Oracle de las rutas de acceso es heurística. Si existe más de una forma de ejecutar una sentencia SQL, entonces el RBO siempre utiliza la operación con la menor categoría. Usualmente, las operaciones de menor categoría se ejecutan más rápido que aquellas asociadas con conceptos de una categoría mayor.

El RBO es elegido automáticamente si todas las condiciones siguientes aplican:

- Oracle versión 6 o menor.
- OPTIMIZER_MODE = CHOOSE (Oracle 7 a Oracle 9i).
  - No existen estadísticas.
  - No se añaden hints a las sentencias SQL.
- ALTER SESSION SET OPTIMIZER_MODE = RULE (de Oracle 7 a Oracle 10g).
- Utilizando el hint: SELECT /*+ RULE */... (de Oracle 7 a Oracle 10g). [32]

RBO Ruta 1: Un solo registro por ROWID. Single Row by ROWID.
Esta ruta de acceso está disponible únicamente si la cláusula WHERE de la sentencia identifica los registros seleccionados por ROWID. Para ejecutar la sentencia, Oracle accede a la tabla por ROWID.
RBO Ruta 2: Un único registro por reunión cluster. Single Row by Cluster Join.
RBO Ruta 3: Un único registro por llave cluster de asociación con llave primaria o única. Single Row by Hash Cluster Key with Unique or Primary Key.

RBO Ruta 4: Un único registro por llave primaria o única. Single Row by Unique or Primary Key.
Esta ruta de acceso está disponible si la cláusula WHERE de la sentencia utiliza todas las columnas de una llave primaria o única en condiciones de igualdad. Para llaves compuestas, las condiciones de igualdad deben combinarse con operadores AND. Para ejecutar la sentencia, Oracle realiza una lectura única en el índice de la llave primaria o única para recuperar un único ROWID, y entonces accede a la tabla por ese ROWID.
RBO Ruta 5: Reunión por agrupación. Clustered Join.
RBO Ruta 6: Llave cluster de asociación. Hash Cluster Key.
RBO Ruta 7: Llave cluster indexada. Indexed Cluster Key.
RBO Ruta 8: Índice compuesto. Composite Index.
Esta ruta de acceso está disponible si la cláusula WHERE de la sentencia utiliza todas las columnas de un índice compuesto en condiciones de igualdad combinadas con operadores AND. Para ejecutar la sentencia, Oracle realiza una lectura de rango en el índice para recuperar ROWIDs de los registros seleccionados y entonces accede la tabla por esos ROWIDs.
RBO Ruta 9: Índices de una sola columna. Single-Column Indexes.
Esta ruta de acceso está disponible si la cláusula WHERE de la sentencia utiliza las columnas de uno o más índices de una sola columna en condiciones de igualdad. Para varios índices de una sola columna, las condiciones deben combinarse con operadores AND. Si la cláusula WHERE utiliza la columna de únicamente un índice, entonces Oracle ejecuta la sentencia realizando una lectura de rango en el índice para recuperar los ROWIDs de los registros seleccionados y entonces accede la tabla por esos ROWIDs. Si la cláusula WHERE utiliza columnas de varios índices de una sola columna, entonces Oracle ejecuta la sentencia realizando una lectura de rango en cada índice para recuperar los ROWIDs de los registros que satisfacen cada condición. Oracle entonces mezcla los conjuntos de ROWIDs para obtener un conjunto de ROWIDs de registros que satisfacen todas las condiciones. Oracle entonces accede la tabla utilizando esos ROWIDs Oracle puede mezclar hasta cinco índices. Si la cláusula WHERE utiliza columnas de más de cinco índices de una sola columna, Oracle mezcla entonces cinco de ellos, accede la tabla por ROWID y prueba los registros resultantes para determinar si satisfacen las condiciones restantes antes de devolverlos.
RBO Ruta 10: Búsqueda por rangos limitada en columnas indexadas. Bounded Range Search on Indexed Columns.
Esta ruta de acceso está disponible si la cláusula WHERE de la sentencia contiene una condición que utiliza ya sea la columna de un índice de una sola columna o una o más columnas que forman la porción inicial de un índice compuesto. columna = expr columna >[=] expr AND columna <[=] expr columna BETWEEN expr AND expr columna LIKE ‘c%’
Cada una de estas condiciones especifican un rango limitado de valores indexados que son accedidos por la sentencia. Se dice que el rango es limitado porque las condiciones especifican su valor menor y mayor. Para ejecutar la sentencia, Oracle realiza una lectura de rango en el índice, y entonces accede la tabla por ROWID. Esta ruta de acceso no está disponible si la expresión expr referencia la columna indexada.
RBO Ruta 11: Búsqueda por rangos ilimitada en columnas indexadas. Unbounded Range Search on Indexed Columns.
Esta ruta de acceso está disponible si la cláusula WHERE de la sentencia contiene una de las siguientes condiciones que utilizan ya sea la columna de un índice de una sola columna, o una o más columnas de la porción inicial de un índice compuesto. WHERE columna >[=] expr WHERE columna <[=] expr Cada una de estas condiciones especifican un rango ilimitado de los valores de índice accedidos por la sentencia. Se dice que el rango es ilimitado, ya que la condición especifica ya sea su valor mínimo o máximo pero no ambos. Para ejecutar tal sentencia, Oracle realiza una lectura de rango en el índice, y entonces accede la tabla por ROWID.
RBO Ruta 12: Reunión por mezcla. Sort Merge Join.

Esta ruta de acceso está disponible para sentencias que reúnen tablas que no están almacenadas juntas en un cluster, si la cláusula WHERE de la sentencia utiliza columnas de cada tabla en condiciones de igualdad. Para ejecutar tal sentencia, Oracle utiliza una operación de reunión por mezcla. Oracle puede también utilizar una operación en bucle anidado para ejecutar una sentencia de reunión.

RBO Ruta 13: MAX o MIN en columna indexada. MAX or MIN of Indexed Column.

Esta ruta de acceso está disponible para una sentencia SELECT y si todas las condiciones siguientes son verdaderas:

La consulta utiliza la función MAX o MIN para seleccionar el valor máximo o mínimo, ya sea de la columna de un índice de una sola columna o de la columna inicial de un índice compuesto. El índice no puede ser un índice cluster.

El argumento para la función MAX o MIN puede ser cualquier expresión involucrando la columna, una constante, la operación adición, la operación concatenación, o la función CONCAT.

No hay otras expresiones en la lista de selección.

La sentencia no tiene cláusula WHERE o GROUP BY.

Para ejecutar la consulta, Oracle realiza una lectura completa del índice para encontrar el valor indexado máximo o mínimo. Ya que solamente este valor es seleccionado, Oracle no necesita acceder a la tabla después de leer el índice.

RBO Ruta 14: ORDER BY en columna indexada. ORDER BY on Indexed Column.

Esta ruta de acceso está disponible para una sentencia SELECT y si todas las condiciones siguientes son verdaderas:

La consulta contiene una cláusula ORDER BY que utiliza la columna de un índice de una sola columna o la porción inicial de un índice compuesto. El índice no puede ser un índice cluster.

Existe una restricción de integridad de PRIMARY KEY o NOT NULL que garantiza que al menos una de las columnas indexadas listadas en la cláusula ORDER BY no contiene nulos.

El parámetro NLS_SORT está establecido a BINARY.

Para ejecutar la consulta, Oracle realiza una lectura de rango del índice para recuperar ROWIDs de los registros seleccionados en orden de clasificación. Oracle entonces accede la tabla por estos ROWIDs.

RBO Ruta 15: Lectura completa de tabla. Full Table Scan.

Esta ruta de acceso está disponible para cualquier sentencia SQL, sin importar las condiciones de su cláusula WHERE, excepto cuando su cláusula FROM contiene SAMPLE o SAMPLE BLOCK.

La lectura completa de tabla está clasificada como la más baja ruta de acceso en la lista. Esto significa que el RBO siempre elige una ruta de acceso que utiliza un índice si alguno está disponible, aún si una lectura completa de tabla se pudiese ejecutar más rápido.

Las siguientes condiciones hacen las rutas de acceso de índice no disponibles:

columna1 > columna2

columna1 < columna2

columna1 >= columna2

columna1 <= columna2

Donde columna1 y columna2 están en la misma tabla.

columna IS NULL

columna IS NOT NULL

columna NOT IN

columna != expr

columna LIKE '%pattern'

Sin importar si columna está indexada.

expr = expr2

Donde expr es una expresión que opera en una columna con un operador o función, sin importar si la columna está indexada.

NOT EXISTS subquery

pseudocolumna ROWNUM en una vista.

Cualquier condición involucrando una columna que no está indexada.

Cualquier sentencia SQL que contenga solamente estos patrones y ningún otro que haga disponible las rutas de acceso de índices debe utilizar una lectura completa de tabla.

Tabla 2.11 Rutas de acceso para el RBO y su clasificación. [33]

2.12.7 Síntesis de las características esenciales del CBO y RBO.

Optimizador Basado en Costos.	Optimizador Basado en Reglas.
El CBO se introdujo en Oracle 7 y fue creado para proporcionar una alternativa más sofisticada a la optimización basada en reglas. Oracle reconoció que la optimización de SQL podría ser más efectiva si el optimizador estaba al tanto de detalles sobre los datos en las tablas e índices. [34]	El primer y más antiguo modo de optimización. [35]
El optimizador basado en costos utiliza estadísticas sobre las tablas para determinar inteligentemente la forma más eficiente de dar servicio a la consulta SQL. [36]	Oracle utiliza heurísticas desde el diccionario de datos para determinar la forma más eficiente de dar servicio a una consulta y traducir el comando declarativo SQL en un plan de navegación real para extraer los datos. [35]
El CBO tiene disponibles rutas de acceso avanzadas y planes de consulta sofisticados. [37]	El RBO es fácil y simple de entender. Su principal filosofía es "si existe un índice, utilízalo", por lo tanto el RBO no es muy bueno con grandes reportes o en ambientes de data warehouse. El RBO examina únicamente la disponibilidad de ciertas rutas de acceso. Las rutas de acceso están clasificadas y la ruta con la mejor categoría es utilizada. [38]
Los desarrolladores no tienen que saber como escribir la mejor consulta en todos los casos. [39] El optimizador basado en costos determina automáticamente la ruta de ejecución más eficiente, y el programador es dotado con hints que pueden añadirse a la consulta para alterar la ruta de acceso. [35]	La indexación de las tablas y el orden de las cláusulas dentro de la sentencia SQL controlan la ruta de acceso en la optimización basada en reglas. [35]
El CBO no es un producto terminado, el CBO está cambiando continuamente entre versiones. [38]	El RBO ha permanecido más o menos igual desde Oracle 5.1.22 hasta 9.2. Esto representa más de una década sin cambio. [38]

2.13 Reuniendo Estadísticas.

Se pueden generar estadísticas que cuantifican la distribución de los datos y características de almacenamiento de tablas, columnas, índices y particiones.

El enfoque de optimización basado en costos utiliza estas estadísticas para calcular la selectividad de predicados y para estimar el costo de cada plan de ejecución.

La selectividad es la fracción de registros en una tabla que el predicado de una sentencia SQL elige. El optimizador utiliza la selectividad de un predicado

para estimar el costo de un método de acceso particular y para determinar el orden de reunión óptimo y el método de reunión. [40]

Condiciones de igualdad.	Llave única o primaria = constante.	Es un predicado de un único registro, devolviendo a lo mucho un registro, tiene la selectividad máxima.
	Índice no único = constante.	La consulta puede devolver más de un registro, así que el optimizador utiliza el número de valores distintos en la columna. La selectividad es 1 dividido por el número de valores distintos.
Rangos limitados y sin límites.		Requieren estadísticas para calcular la selectividad. El CBO utiliza la siguiente fórmula: $\frac{alto - bajo + 1}{máximo - mínimo + 1}$

Tabla 2.12 Reglas del optimizador para el cálculo de selectividad de predicados. [41]

Las estadísticas son almacenadas en el diccionario de datos y pueden ser exportadas de una base de datos e importadas en otra.

Se deberían reunir estadísticas periódicamente para objetos cuyas estadísticas se vuelven anticuadas a través del tiempo debido a los volúmenes de datos cambiantes o cambios en los valores de columnas. Las nuevas estadísticas deberían reunirse después de que los datos o estructuras de los objetos del esquema son modificados de forma que vuelven las estadísticas anteriores inexactas.

Antes de Oracle 9i, las estadísticas eran calculadas con la sentencia ANALYZE. En Oracle 9i, el método preferido de calcular estadísticas es con el paquete DBMS_STATS, pero el comando ANALYZE contiene todavía esta funcionalidad. En Oracle 10g, la única forma de calcular estadísticas es con el paquete DBMS_STATS. Oracle ha detenido el desarrollo en el comando ANALYZE para calcular estadísticas. [42]

Las estadísticas generadas incluyen lo siguiente:

USER_TABLES, USER_TAB_STATISTICS – estadísticas en tablas	
NUM_ROWS	Número de registros
BLOCKS	Número de bloques de datos utilizados
EMPTY_BLOCKS	Número de bloques de datos no utilizados
AVG_SPACE	Espacio libre promedio en bytes disponible por bloque
AVG_ROW_LEN	Longitud de registro promedio en bytes
LAST_ANALYZED	Fecha en que las estadísticas fueron más recientemente calculadas
SAMPLE_SIZE	Tamaño de muestra empleado al calcular estadísticas
GLOBAL_STATS	Indica si las estadísticas fueron colectadas para una tabla particionada completa o no
USER_STATS	Indica si las estadísticas fueron proporcionadas por el usuario o calculadas por el sistema

USER_TAB_COLUMNS, USER_TAB_COL_STATISTICS – estadísticas en columnas	
NUM_DISTINCT	Número de valores distintos por columna
LOW_VALUE	Valor mínimo de columna
HIGH_VALUE	Valor máximo de columna
DENSITY	Factor de densidad de columna (1/NUM_DISTINCT)
NUM_NULLS	Número de valores nulos
NUM_BUCKETS	Número de cubetas de histograma
LAST_ANALYZED	Fecha en que las estadísticas fueron más recientemente calculadas
SAMPLE_SIZE	Tamaño de muestra empleado al calcular estadísticas
GLOBAL_STATS	Indica si las estadísticas fueron colectadas para una tabla particionada completa o no
USER_STATS	Indica si las estadísticas fueron proporcionadas por el usuario o calculadas por el sistema
AVG_COL_LEN	Longitud promedio de columna en bytes
HISTOGRAM	Indica la presencia o tipo de histograma (NONE | FREQUENCY | HEIGHT BALANCED)

USER_INDEXES, USER_IND_STATISTICS – estadísticas en índices	
BLEVEL	Profundidad del árbol de índice. Si es cero, entonces el bloque raíz es el único nodo hoja en el árbol
LEAF_BLOCKS	Número de bloques hoja
DISTINCT_KEYS	Número de llaves distintas
AVG_LEAF_BLOCKS_PER_KEY	Número promedio de bloques hoja por llave
AVG_DATA_BLOCKS_PER_KEY	Número promedio de bloques de datos por llave
CLUSTERING_FACTOR	Indica el orden relativo de registros de datos en la tabla Si el valor es cercano al número de bloques en la tabla, los registros están ordenados Si el valor es cercano al número de registros en la tabla, los registros están ordenados al azar. En tal caso, es poco probable que las entradas del índice en el mismo bloque hoja apuntarán a registros en el mismo bloque de datos
LAST_ANALYZED	Fecha en que las estadísticas fueron más recientemente calculadas
SAMPLE_SIZE	Tamaño de muestra empleado al calcular estadísticas
USER_STATS	Indica si las estadísticas fueron proporcionadas por el usuario o calculadas por el sistema
GLOBAL_STATS	Indica si las estadísticas fueron colectadas para un índice particionado completo o no

Tabla 2.13 Principales estadísticas generadas para el CBO con el paquete DBMS_STATS.

2.13.1 Generando estadísticas.

2.13.2 Reunión manual de estadísticas.

La base de datos Oracle genera estadísticas utilizando las siguientes técnicas:

- Estimación basada en muestra de datos aleatoria. Se puede especificar el porcentaje de muestreo; y si debiese estar basado en registros o bloques.
- Cálculo exacto.
- Métodos de recolección de estadísticas definidos por el usuario.

Cuando se generan estadísticas para una tabla, columna o índice, si el diccionario de datos contiene estadísticas para el objeto, Oracle actualiza las estadísticas existentes. Oracle también invalida cualquier sentencia SQL actualmente analizada que acceda al objeto. La siguiente vez que tal sentencia se ejecuta, el optimizador automáticamente elige un nuevo plan de ejecución basado en las nuevas estadísticas. [43]

2.13.3 El paquete DBMS_STATS.

Este paquete permite generar y administrar estadísticas para la optimización basada en costos. Se puede utilizar este paquete para reunir, modificar, ver, exportar, importar y eliminar estadísticas. También para identificar o nombrar estadísticas reunidas.

El paquete DBMS_STATS puede reunir estadísticas en índices, tablas, columnas, y particiones, así como también estadísticas de todos los objetos en un esquema o la base de datos.

Oracle recomienda establecer el parámetro ESTIMATE_PERCENT de los procedimientos de reunión del paquete DBMS_STATS a DBMS_STATS.AUTO_SAMPLE_SIZE, lo que le permite a Oracle determinar el mejor tamaño de muestra para buenas estadísticas. [43]

2.13.3.1 Reuniendo estadísticas de tabla, índice y columna.

El procedimiento DBMS_STATS.GATHER_TABLE_STATS reúne estadísticas de tabla, columna e índice.

Para verificar que las estadísticas de una tabla están disponibles se consulta la vista del diccionario de datos USER_TABLES.

El procedimiento DBMS_STATS.GATHER_INDEX_STATS se puede utilizar también para recolectar estadísticas de índices.

Para verificar que las estadísticas de índice están disponibles y decidir cuáles son los mejores índices a utilizar en una aplicación, se consulta la vista del diccionario de datos USER_INDEXES.

Para verificar que las estadísticas de columna están disponibles se consulta la vista del diccionario de datos USER_TAB_COL_STATISTICS. [44]

2.13.4 Reunión automática de estadísticas.

La forma recomendada para reunir estadísticas es permitir que Oracle reúna automáticamente las estadísticas. Oracle reúne estadísticas en todos los objetos de la base de datos automáticamente y mantiene esas estadísticas en una tarea de mantenimiento programada regularmente. La recolección automatizada de estadísticas elimina muchas de las tareas manuales asociadas con la gestión del optimizador de consultas, y reduce significativamente las posibilidades de obtener planes de ejecución pobres debido a estadísticas faltantes o anticuadas.

Las estadísticas del optimizador son reunidas automáticamente con la tarea GATHER_STATS_JOB. Esta tarea reúne estadísticas en todos los objetos de la base de datos que tienen estadísticas faltantes o anticuadas. Esta tarea se crea automáticamente al momento de creación de la base de datos y es administrada por el programador automático de tareas. El programador automático de tareas ejecuta esta tarea cuando la ventana de mantenimiento se abre. Por defecto, la ventana de mantenimiento abre cada noche de 10 PM a 6 AM y todo el día en fines de semana.

El atributo stop_on_window_close controla si la tarea GATHER_STATS_JOB continúa cuando la ventana de mantenimiento cierra. La configuración por defecto para el atributo stop_on_window_close es TRUE, ocasionando que el programador automático de tareas termine la tarea

GATHER_STATS_JOB cuando la ventana de mantenimiento cierra. Los objetos restantes son procesados entonces en la siguiente ventana de mantenimiento.

La tarea GATHER_STATS_JOB reúne estadísticas llamando al procedimiento DBMS_STATS.GATHER_DATABASE_STATS_JOB_PROC. Este procedimiento da prioridad a los objetos de base de datos que requieren estadísticas de forma que sean procesados primero. Estos objetos son objetos de la base de datos sin estadísticas reunidas previamente o con estadísticas anticuadas porque el objeto subyacente ha sido modificado significativamente (más del 10% de registros). [45]

Ejecutando la siguiente consulta sobre la vista DBA_SCHEDULER_JOBS se observan algunos detalles de la tarea GATHER_STATS_JOB:

```
SELECT job_name, enabled, last_start_date, stop_on_window_close, comments
  FROM dba_scheduler_jobs
 WHERE job_name = 'GATHER_STATS_JOB';
```

JOB_NAME	ENABLED	LAST_START_DATE	STOP_ON_WINDOW_CLOSE	COMMENTS
GATHER_STATS_JOB	TRUE	3/25/2009 10:01:32.370362 PM -04:00	TRUE	optimizer statistics collection

Tabla 2.14 Consultando la tarea GATHER_STATS_JOB en la vista DBA_SCHEDULER_JOBS.

2.13.5 Histogramas.

Cuando se reúnen estadísticas en una tabla, el paquete DBMS_STATS reúne información sobre la distribución de datos de las columnas dentro de la tabla. La información más básica sobre la distribución de datos es el valor máximo y mínimo de la columna. Sin embargo, este nivel de estadísticas puede ser insuficiente para las necesidades del optimizador, si los datos dentro de la columna son sesgados. Para distribuciones de datos sesgadas, pueden crearse también histogramas como parte de las estadísticas de columna para describir la distribución de los datos de una columna dada. [46]

Un histograma divide los valores en la columna en grupos, de forma que todos los valores de columna en un grupo están dentro del mismo rango. Los histogramas proporcionan mejores estimados de selectividad en la presencia de sesgo de datos, resultando en planes de ejecución óptimos con distribuciones de datos no uniformes. [47]

2.13.5.1 Cuando utilizar histogramas.

Los histogramas pueden afectar el desempeño y deberían utilizarse únicamente cuando mejoran sustancialmente los planes de una consulta. Los datos estadísticos de un histograma son persistentes, así que el espacio requerido para guardar los datos depende del tamaño de la muestra. En general, se deben crear histogramas en columnas que se utilizan frecuentemente en la cláusula WHERE de consultas y que tienen una distribución de datos altamente sesgada. Para datos distribuidos uniformemente, el optimizador puede hacer suposiciones bastante precisas sobre el costo de ejecutar una sentencia particular sin el uso de histogramas. [48]

Por defecto en Oracle 10g se utiliza la opción FOR ALL COLUMNS SIZE AUTO al reunir estadísticas de columna. Con esta configuración, Oracle automáticamente determina cuales columnas requieren histogramas y el número de cubetas (tamaño) de cada histograma. [49]

2.13.6 Optimizando sentencias SQL.

La afinación de sentencias SQL consiste en encontrar formas más eficientes de procesar la misma carga de trabajo, ya que es posible cambiar el plan de ejecución de una sentencia sin alterar la funcionalidad para disminuir el consumo de recursos. [50]

2.13.6.1 Identificación y reunión de datos en sentencias SQL que utilizan muchos recursos.

El primer paso es identificar las sentencias SQL problemáticas, sentencias que se están desempeñando pobremente, que están utilizando mayores recursos e impactan el desempeño de la base de datos Oracle. [51]

La siguiente fase es reunir información necesaria para examinar las sentencias y afinarlas.

La información reunida incluye lo siguiente:

1. Texto SQL completo.
2. Estructura de las tablas referenciadas en la sentencia SQL.

3. Definiciones de cualquiera índices (columnas, ordenación de columnas), y si los índices son únicos o no únicos.

4. Estadísticas del CBO para los segmentos (incluyendo el número de registros de cada tabla, selectividad de las columnas indexadas), incluyendo la fecha en que los segmentos fueron analizados por última vez.

5. Definiciones de cualquier vista referida en la sentencia SQL.

6. Repetir los pasos dos, tres y cuatro por cualquier tabla referenciada en las definiciones de vista encontradas en el paso cinco.

7. Plan del optimizador para la sentencia SQL.

8. Cualquiera planes previos del optimizador para esa sentencia. [52]

2.13.6.2 Lineamientos para desarrollar sentencias SQL eficientes.

- Verificar las estadísticas del optimizador.

  El CBO determina el plan óptimo de ejecución de cada sentencia SQL utilizando estadísticas reunidas en tablas e índices. Si estas estadísticas no han sido reunidas, o si las estadísticas ya no son representativas de los datos almacenados en la base de datos, entonces el optimizador no tiene información suficiente para generar el mejor plan.

- Revisar el plan de ejecución.

  El objetivo es comprobar si las rutas de acceso son las óptimas.

  Si alguna de estas condiciones no es la mejor, se debe entonces considerar reestructurar la sentencia SQL o los índices disponibles en las tablas.

  - La tabla que conduce tiene el mejor filtro, el filtro que elimina el mayor porcentaje de la tabla.
  - El orden de reunión en cada paso significa que el menor número de registros es devuelto al siguiente paso.
  - El método de reunión es apropiado para el número de registros que son devueltos.
  - Las vistas son usadas eficientemente.
  - No hay productos cartesianos accidentales.

- Cada tabla es accedida eficientemente. Tener en cuenta los predicados en la sentencia SQL y el número de registros en la tabla. Buscar actividad sospechosa, como lecturas completas de tabla en tablas con un gran número de registros, que tienen predicados en la cláusula WHERE.

- Reestructurar las sentencias SQL.

  Rescribir una sentencia SQL ineficiente es a menudo más fácil que repararla. Si se entiende el propósito de una sentencia dada, entonces se puede ser capaz de escribir fácil y rápidamente una nueva sentencia que cumpla el requerimiento.

  - Utilizar equirreuniones siempre que sea posible.
  - Evitar columnas transformadas mediante funciones SQL, expresiones complejas y conversiones de tipo implícitas en la cláusula WHERE. Estas expresiones previenen al optimizador de asignar estimados de cardinalidad o selectividad y pueden afectar a su vez el plan global y el método de reunión.
  - Escribir sentencias SQL separadas para tareas específicas. Si se quiere que SQL haga cosas diferentes, es mejor entonces escribir varias sentencias, en vez de escribir una sola sentencia para hacer diferentes cosas.

    El objetivo principal de la afinación de SQL es evitar realizar trabajo innecesario para acceder registros que no afectan el resultado.

- Reestructurar los índices. Esto puede involucrar lo siguiente:
  - Eliminar índices no discriminativos para acelerar el DML.
  - Indexar rutas de acceso de desempeño crítico.
  - Considerar reordenar columnas en índices concatenados existentes.
  - Agregar columnas al índice para mejorar la selectividad. [53]

2.13.7 Afinación de SQL en Oracle 10g.

En Oracle 10g, el proceso de afinación de SQL ha sido automatizado introduciendo una nueva función de afinación agregada del optimizador de

consultas llamada Optimizador de Afinación Automático. Este nuevo modo del optimizador de consultas realiza análisis adicionales durante el proceso de construcción del plan de ejecución para una sentencia SQL. En contraste, el modo normal del optimizador tiene restricciones rígidas en la cantidad de tiempo y recursos de sistema que puede utilizar para encontrar un buen plan de ejecución para una sentencia SQL dada.

El Optimizador de Afinación Automático realiza cuatro tipos de análisis durante el proceso de generación del plan:

- Análisis de estadísticas: El Optimizador de Afinación Automático verifica cada objeto de la consulta por estadísticas faltantes o anticuadas y hace la recomendación de reunir estadísticas relevantes.
- Descripción de SQL: El Optimizador de Afinación Automático verifica sus propios estimados y colecta información auxiliar para eliminar errores de estimación. Colecta también información auxiliar en la forma de configuraciones personalizadas del optimizador basado en los registros de ejecución anteriores de la sentencia SQL. Construye una descripción de SQL empleando la información auxiliar y hace la recomendación de crearla, lo cual permite al optimizador de consultas (bajo el modo normal) generar un plan bien afinado para la correspondiente sentencia SQL.
- Análisis de ruta de acceso: El Optimizador de Afinación Automático explora si un nuevo índice puede utilizarse para mejorar de modo significativo el acceso a cada tabla en la consulta, y cuando es apropiado hace la recomendación para crear tales índices.
- Análisis de estructura SQL: El Optimizador de Afinación Automático trata de identificar sentencias SQL propicias a malos planes, y hace sugerencias relevantes para reestructurarlas. Las reestructuraciones sugeridas pueden ser cambios sintácticos y semánticos al código SQL.

Las funciones de afinación de SQL automático son expuestas a través de una herramienta del servidor llamada SQL Tuning Advisor. Esta herramienta toma una o más sentencias SQL como entrada e invoca al Optimizador de Afinación Automático para llevar a cabo la afinación de SQL en las sentencias. [54]

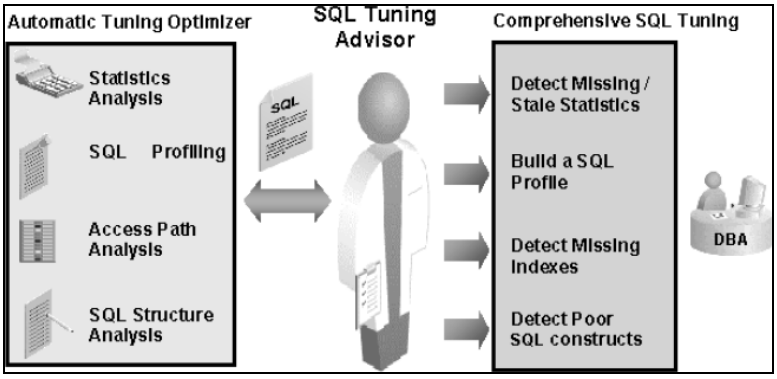

Figura 2.8 Arquitectura de Afinación de SQL Automático.

SQL Tuning Advisor proporciona opciones para administrar el alcance y duración de una tarea de afinación. El alcance de una tarea de afinación puede ser establecido a LIMITED o COMPREHENSIVE.

Con la opción LIMITED, SQL Tuning Advisor produce recomendaciones basado en verificaciones de estadísticas, análisis de rutas de acceso, y análisis de estructura SQL.

Si se elige la opción COMPREHENSIVE, SQL Tuning Advisor lleva a cabo todo el análisis que realiza bajo el alcance LIMITED más la descripción de SQL. Con la opción COMPREHENSIVE se puede también especificar un límite de tiempo para la tarea de afinación, que por defecto es 30 minutos.

Después de analizar las sentencias SQL, SQL Tuning Advisor proporciona recomendaciones en optimizar el plan de ejecución, el fundamento para la optimización propuesta, el beneficio de desempeño esperado, y el comando para implementar la sugerencia. Simplemente se tiene que elegir si aceptar o no las recomendaciones para optimizar las sentencias SQL.

Se puede ejecutar SQL Tuning Advisor utilizando los procedimientos en el paquete DBMS_SQLTUNE. DBMS_SQLTUNE es un nuevo paquete añadido en Oracle 10g y contiene las APIs necesarias para realizar afinación automática de sentencias, y administrar descripciones de SQL y conjuntos de afinación SQL.

Para utilizar las APIs el usuario debe ser otorgado de privilegios específicos. El uso de los procedimientos de afinación de SQL incluyendo la creación de tareas de afinación requieren el privilegio ADVISOR. [55]

Realizar la afinación automática de SQL utilizando el paquete DBMS_SQLTUNE es un proceso de varios pasos:

1. Crear un conjunto de afinación SQL si se afinarán varias sentencias.
2. Crear una tarea de afinación SQL.
3. Ejecutar una tarea de afinación SQL.
4. Desplegar los resultados de una tarea de afinación.
5. Implementar recomendaciones como sea apropiado. [56]

3 METODOLOGÍA

3.1 Creación de un caso práctico que permita la asimilación del problema de optimización de consultas.

Se diseñó el siguiente caso práctico para permitir:

- Asimilar el problema de optimización de consultas.
- Mostrar las características del optimizador basado en costos y del optimizador basado en reglas de un manejador de base de datos relacional.
- Conocer algunas herramientas de optimización que acompañan a un manejador de base de datos relacional.

3.1.1 El servidor de base de datos.

Teniendo un servidor de base de datos Oracle 10g, específicamente *Oracle Database 10g Enterprise Edition Release 10.2.0.3.0 - 64bit Production* con los parámetros de inicio que se muestran en la siguiente tabla.

	Nombre del parámetro	Valor del parámetro en un formato amigable	Indica si el parámetro está establecido al valor por defecto (TRUE) o el valor del parámetro fue especificado en el archivo de parámetros (FALSE)	Indica si el parámetro puede cambiarse con ALTER SESSION (TRUE) o no (FALSE)	Descripción del parámetro
	NAME	**DISPLAY_VALUE**	**ISDEFAULT**	**ISSES_MODIFIABLE**	**DESCRIPTION**
Administración automática de memoria compartida.	db_cache_size	0	TRUE	FALSE	Size of DEFAULT buffer pool for standard block size buffers
	java_pool_size	0	TRUE	FALSE	size in bytes of java pool
	large_pool_size	0	TRUE	FALSE	size in bytes of large pool
	sga_target	500M	FALSE	FALSE	Target size of SGA
	shared_pool_size	0	TRUE	FALSE	size in bytes of shared pool
	statistics_level	TYPICAL	TRUE	TRUE	statistics level
	streams_pool_size	0	TRUE	FALSE	size in bytes of the streams pool
Administración automática de memoria PGA	hash_area_size	131072	TRUE	TRUE	size of in-memory hash work area
	pga_aggregate_target	150M	FALSE	FALSE	Target size for the aggregate PGA memory consumed by the instance
	sort_area_size	65536	TRUE	TRUE	size of in-memory sort work area
	workarea_size_policy	AUTO	TRUE	TRUE	policy used to size SQL working areas (MANUAL/AUTO)
Parámetros para habilitar características y controlar el comportamiento del CBO.	cursor_sharing	EXACT	TRUE	TRUE	cursor sharing mode
	db_block_size	8192	FALSE	FALSE	Size of database block in bytes
	db_file_multiblock_read_count	16	FALSE	TRUE	db block to be read each IO
	optimizer_mode	ALL_ROWS	TRUE	TRUE	optimizer mode

Tabla 3.1 Parámetros de inicio de la vista V$PARAMETER que afectan el optimizador de Oracle.

Siendo un servidor de base de datos para aplicaciones On-Line Transaction Processing, se observa que el modo por defecto del optimizador es ALL_ROWS,

lo que significa que el optimizador utiliza un enfoque basado en costos para todas las sentencias SQL y optimiza con el objetivo de mejor procesamiento. También que la administración automática de memoria compartida y de memoria PGA están habilitadas.

3.1.2 Diagrama entidad-relación de la base de datos.

Se tiene el esquema de base de datos TBGEX de la siguiente figura:

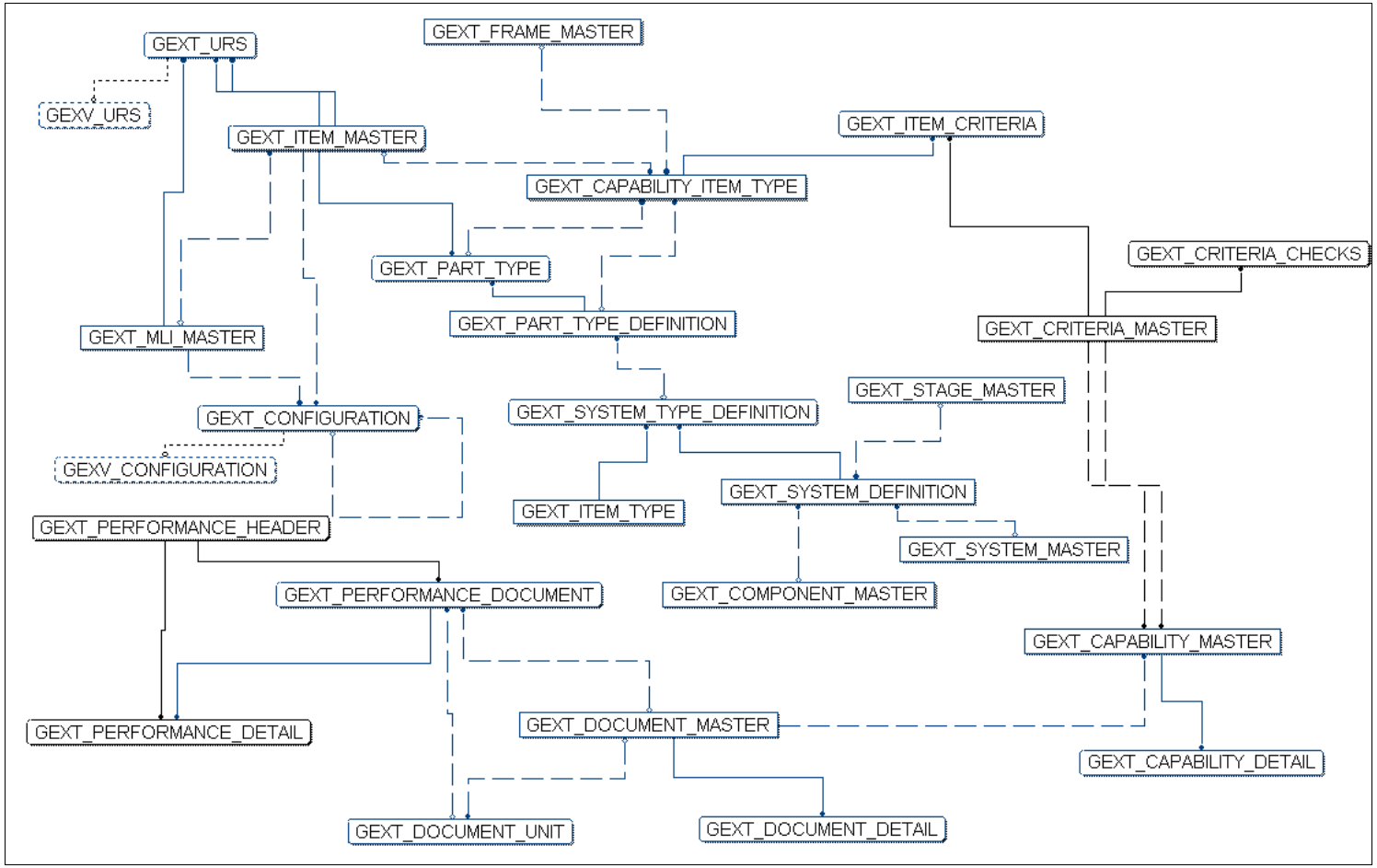

Figura 3.1 Diagrama Entidad-Relación del esquema TBGEX.

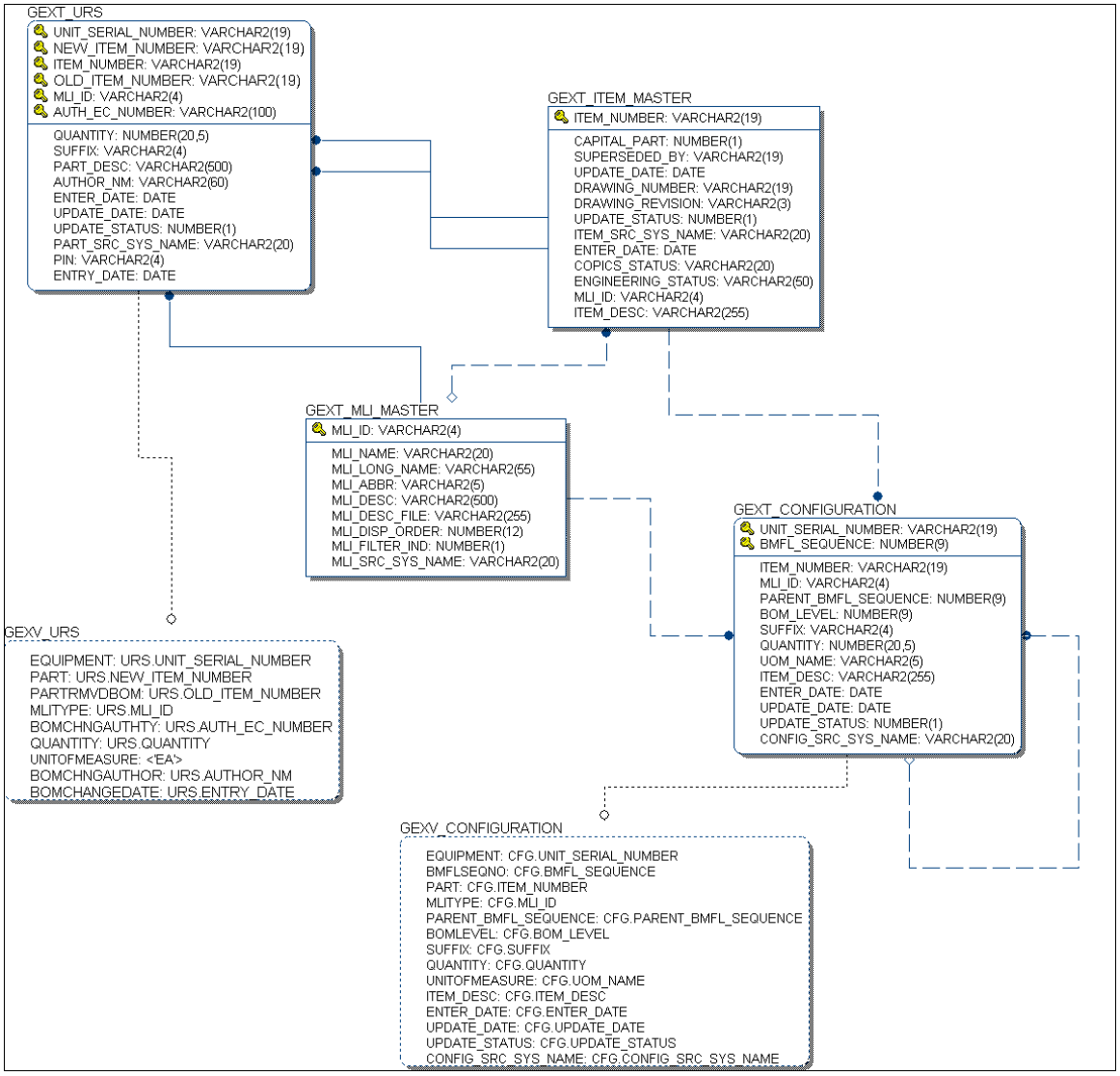

Figura 3.2 Diagrama Entidad-Relación del esquema TBGEX: Unit Record Sheet & Configuration.

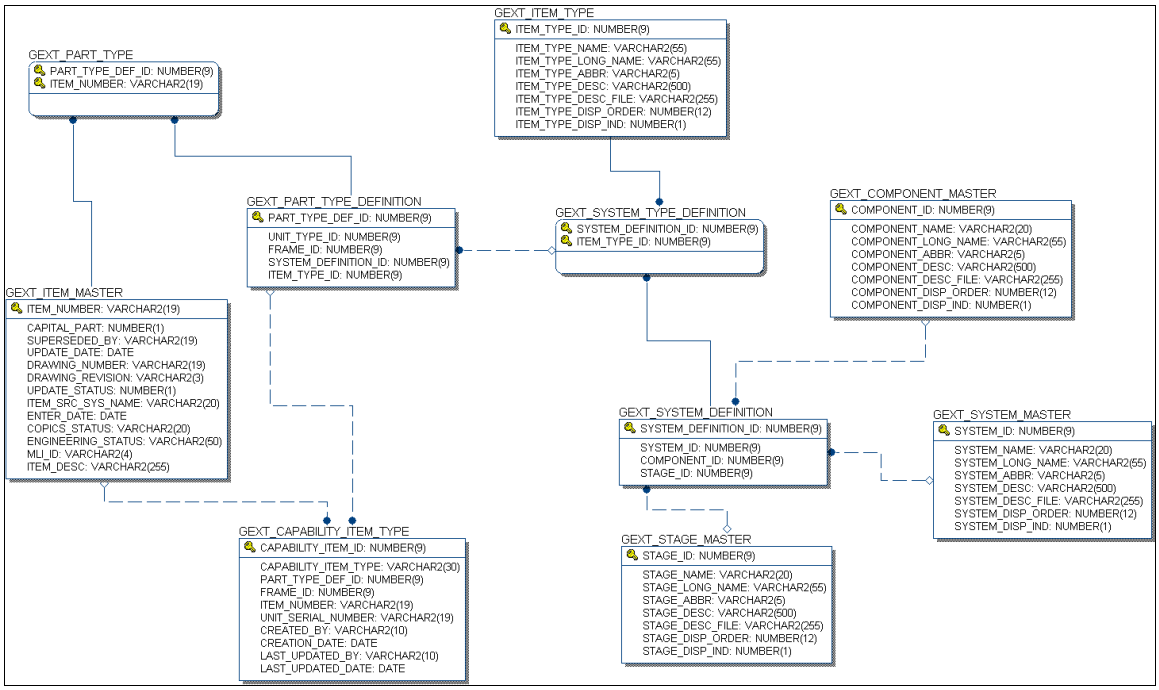

Figura 3.3 Diagrama Entidad-Relación del esquema TBGEX: Part Type Categories.

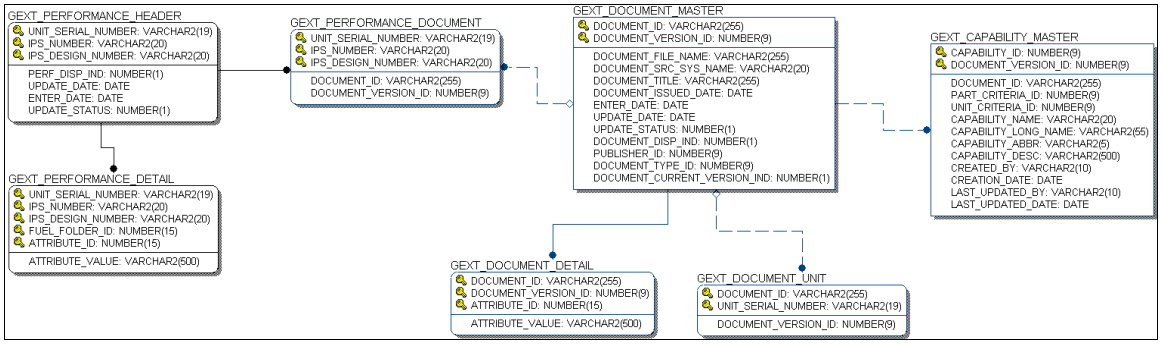

Figura 3.4 Diagrama Entidad-Relación del esquema TBGEX: Document & Performance.

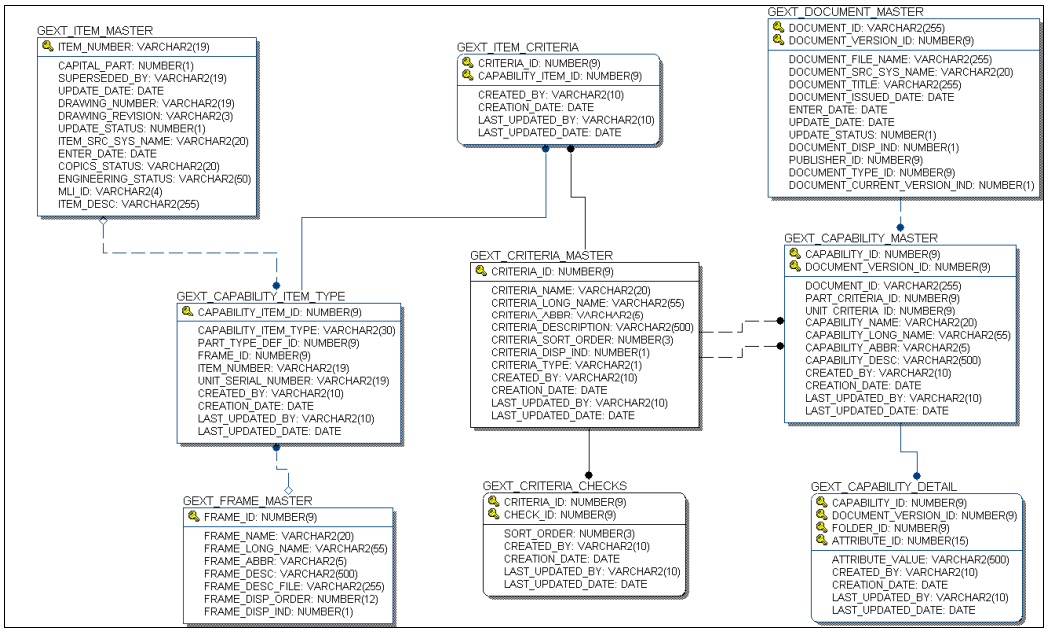

Figura 3.5 Diagrama Entidad-Relación del esquema TBGEX: Maintenance Intervals.

3.1.3 Estadísticas del diccionario de datos.

3.1.3.1 Estadísticas de tablas.

El esquema anterior cuenta con las siguientes estadísticas en el diccionario de datos:

TABLE_NAME	NUM_ROWS	BLOCKS	EMPTY_BLOCKS	AVG_SPACE	AVG_ROW_LEN	LAST_ANALYZED	SAMPLE_SIZE	GLOBAL_STATS	USER_STATS
GEXT_CAPABILITY_DETAIL	22420	149	0	0	51	5/8/2009 16:08	22420	YES	NO
GEXT_CAPABILITY_ITEM_TYPE	1701	14	0	0	45	5/8/2009 16:08	1701	YES	NO
GEXT_CAPABILITY_MASTER	608	9	0	0	105	5/8/2009 16:08	608	YES	NO
GEXT_COMPONENT_MASTER	29	1	0	0	48	5/8/2009 16:08	29	YES	NO
GEXT_CONFIGURATION	34593994	187549	0	0	42	5/8/2009 16:07	48531	YES	NO
GEXT_CRITERIA_CHECKS	974	9	0	0	29	5/8/2009 16:08	974	YES	NO
GEXT_CRITERIA_MASTER	259	3	0	0	75	5/8/2009 16:08	259	YES	NO
GEXT_DOCUMENT_DETAIL	1553064	5580	0	0	33	5/8/2009 16:11	1553064	YES	NO
GEXT_DOCUMENT_MASTER	232207	3633	0	0	111	5/8/2009 16:11	232207	YES	NO
GEXT_DOCUMENT_UNIT	165176	519	0	0	19	5/8/2009 16:11	41294	YES	NO
GEXT_FRAME_MASTER	15	1	0	0	44	5/8/2009 16:11	15	YES	NO
GEXT_ITEM_CRITERIA	453	4	0	0	26	5/8/2009 16:11	453	YES	NO
GEXT_ITEM_MASTER	1835404	30853	0	0	116	5/8/2009 16:05	42792	YES	NO
GEXT_ITEM_TYPE	12	1	0	0	38	5/8/2009 16:12	12	YES	NO
GEXT_MLI_MASTER	8147	59	0	0	44	5/8/2009 16:12	8147	YES	NO
GEXT_PART_TYPE	20466	64	0	0	16	5/8/2009 16:12	20466	YES	NO
GEXT_PART_TYPE_DEFINITION	1523	9	0	0	17	5/8/2009 16:12	1523	YES	NO
GEXT_PERFORMANCE_DETAIL	531922	2698	0	0	42	5/8/2009 16:12	72407	YES	NO
GEXT_PERFORMANCE_DOCUMENT	7057	29	0	0	26	5/8/2009 16:12	7057	YES	NO
GEXT_PERFORMANCE_HEADER	8523	54	0	0	37	5/8/2009 16:12	8523	YES	NO
GEXT_STAGE_MASTER	45	1	0	0	48	5/8/2009 16:12	45	YES	NO
GEXT_SYSTEM_DEFINITION	153	1	0	0	15	5/8/2009 16:12	153	YES	NO
GEXT_SYSTEM_MASTER	6	1	0	0	33	5/8/2009 16:12	6	YES	NO
GEXT_SYSTEM_TYPE_DEFINITION	406	1	0	0	6	5/8/2009 16:12	406	YES	NO
GEXT_URS	268981	3863	0	0	97	5/8/2009 16:13	39130	YES	NO

Tabla 3.2 Estadísticas en tablas del esquema TBGEX de la vista USER_TABLES.

3.1.3.2 Estadísticas de columnas.

TABLE_NAME	COLUMN_NAME	NUM_DISTINCT	LOW_VALUE	HIGH_VALUE	DENSITY	NUM_NULLS	NUM_BUCKETS	LAST_ANALYZED	SAMPLE_SIZE	GLOBAL_STATS	USER_STATS	AVG_COL_LEN	HISTOGRAM
GEXT_CAPABILITY_DETAIL	ATTRIBUTE_ID	7	C3020102	C302010B	0.14285714	0	1	5/8/2009 16:08	5474	YES	NO	5	NONE
GEXT_CAPABILITY_DETAIL	ATTRIBUTE_VALUE	356	28436F6E646974696F6E20426173656420D61696E74656E616E636520666F72	54686572652061726520616464646974696F6E616C2066616C6C6F7574206 2063	0.00280899	2906	1	5/8/2009 16:08	4761	YES	NO	7	NONE
GEXT_CAPABILITY_DETAIL	CAPABILITY_ID	419	C20B02	C21454	0.00238663	0	1	5/8/2009 16:08	22420	YES	NO	4	NONE
GEXT_CAPABILITY_DETAIL	DOCUMENT_VERSION_ID	8	C102	C111	0.12500000	0	1	5/8/2009 16:08	5474	YES	NO	3	NONE
GEXT_CAPABILITY_DETAIL	FOLDER_ID	18	C20202	C20213	0.05555556	0	1	5/8/2009 16:08	5474	YES	NO	4	NONE
GEXT_CAPABILITY_ITEM_TYPE	CAPABILITY_ITEM_ID	1701	C20B02	C23448	0.00058789	0	1	5/8/2009 16:08	1701	YES	NO	4	NONE
GEXT_CAPABILITY_ITEM_TYPE	CAPABILITY_ITEM_TYPE	2	4954454D5F4E554D424552	504152545F545950455F4445465F4944	0.50000000	0	1	5/8/2009 16:08	1701	YES	NO	17	NONE
GEXT_CAPABILITY_ITEM_TYPE	ITEM_NUMBER	171	313033453336323047303235	393739453033353847303031	0.00584795	1530	1	5/8/2009 16:08	171	YES	NO	3	NONE
GEXT_CAPABILITY_MASTER	CAPABILITY_ID	419	C20B02	C21454	0.00238663	0	1	5/8/2009 16:08	608	YES	NO	4	NONE
GEXT_CAPABILITY_MASTER	DOCUMENT_ID	10	30302D383833	30312D3234372E32	0.10000000	0	1	5/8/2009 16:08	608	YES	NO	9	NONE
GEXT_CAPABILITY_MASTER	DOCUMENT_VERSION_ID	8	C102	C111	0.12500000	0	1	5/8/2009 16:08	608	YES	NO	3	NONE
GEXT_CAPABILITY_MASTER	PART_CRITERIA_ID	189	C104	C2135F	0.00529101	0	1	5/8/2009 16:08	608	YES	NO	4	NONE
GEXT_CAPABILITY_MASTER	UNIT_CRITERIA_ID	33	C102	C21339	0.03030303	0	1	5/8/2009 16:08	608	YES	NO	4	NONE
GEXT_COMPONENT_MASTER	COMPONENT_ID	29	3E6466	C2151D	0.03448276	0	1	5/8/2009 16:08	29	YES	NO	4	NONE
GEXT_CONFIGURATION	BMFL_SEQUENCE	10242	C102	C302284F	0.00009764	0	1	5/8/2009 16:07	48531	YES	NO	5	NONE
GEXT_CONFIGURATION	BOM_LEVEL	16	80	C110	0.06250000	0	1	5/8/2009 16:07	4878	YES	NO	3	NONE
GEXT_CONFIGURATION	ITEM_NUMBER	532444	3130314532303832503033	53554D322D374131504 5413138332D33	0.00000188	0	1	5/8/2009 16:07	4878	YES	NO	12	NONE
GEXT_CONFIGURATION	MLI_ID	766	2D2D2D2D	56323031	0.00130548	0	1	5/8/2009 16:07	4878	YES	NO	5	NONE
GEXT_CONFIGURATION	PARENT_BMFL_SEQUENCE	10140	C102	C3022842	0.00009862	7129	1	5/8/2009 16:07	48521	YES	NO	4	NONE
GEXT_CONFIGURATION	QUANTITY	5	C102	C21F1F1F2215	0.20000000	34586866	1	5/8/2009 16:07	10	YES	NO	2	NONE
GEXT_CONFIGURATION	UNIT_SERIAL_NUMBER	4482	313237373437	323938343531	0.00022311	0	1	5/8/2009 16:07	4878	YES	NO	7	NONE
GEXT_CONFIGURATION	UOM_NAME	2	4541	4541	0.50000000	34586866	1	5/8/2009 16:07	10	YES	NO	2	NONE
GEXT_CRITERIA_CHECKS	CHECK_ID	234	C102	C20323	0.00427350	0	1	5/8/2009 16:08	974	YES	NO	4	NONE
GEXT_CRITERIA_CHECKS	CRITERIA_ID	252	C104	C2135F	0.00396825	0	1	5/8/2009 16:08	974	YES	NO	4	NONE
GEXT_CRITERIA_CHECKS	SORT_ORDER	12	C102	C10D	0.08333333	0	1	5/8/2009 16:08	974	YES	NO	3	NONE
GEXT_CRITERIA_MASTER	CRITERIA_ABBR	92	31324B2037	552D353030	0.01086957	0	1	5/8/2009 16:08	259	YES	NO	6	NONE
GEXT_CRITERIA_MASTER	CRITERIA_DESCRIPTION	32	33313053532C20436C6F7468205365616C205374796C6520533153	552D353030204C45204C6F6F70205363616C6C6F7020533342	0.03125000	227	1	5/8/2009 16:08	32	YES	NO	9	NONE
GEXT_CRITERIA_MASTER	CRITERIA_DISP_IND	1	C102	C102	1.00000000	0	1	5/8/2009 16:08	259	YES	NO	3	NONE
GEXT_CRITERIA_MASTER	CRITERIA_ID	259	C102	C2135F	0.00386100	0	1	5/8/2009 16:08	259	YES	NO	4	NONE
GEXT_CRITERIA_MASTER	CRITERIA_LONG_NAME	158	31324B2037323431	552D353030204C45204C6F6F70205363616C6C6F7020533342	0.00632911	38	1	5/8/2009 16:08	221	YES	NO	17	NONE
GEXT_CRITERIA_MASTER	CRITERIA_NAME	189	31324B2037323431	552D353030204C45204C6F6F70205363616C6C6F	0.00529101	0	1	5/8/2009 16:08	259	YES	NO	17	NONE
GEXT_CRITERIA_MASTER	CRITERIA_SORT_ORDER	0			0.00000000	259	0	5/8/2009 16:08		YES	NO	0	NONE

Tabla 3.3 Estadísticas en columnas del esquema TBGEX de la vista USER_TAB_COLUMNS.

TABLE_NAME	COLUMN_NAME	NUM_DISTINCT	LOW_VALUE	HIGH_VALUE	DENSITY	NUM_NULLS	NUM_BUCKETS	LAST_ANALYZED	SAMPLE_SIZE	GLOBAL_STATS	USER_STATS	AVG_COL_LEN	HISTOGRAM
GEXT_DOCUMENT_DETAIL	ATTRIBUTE_ID	75	C21003	C2110E	0.01333333	0	1	5/8/2009 16:11	5420	YES	NO	4	NONE
GEXT_DOCUMENT_DETAIL	ATTRIBUTE_VALUE	2243	2432362C303030	6E756C6C	0.00044583	74057	1	5/8/2009 16:11	5163	YES	NO	8	NONE
GEXT_DOCUMENT_DETAIL	DOCUMENT_ID	129760	302D31	64343933353571	0.00000771	0	1	5/8/2009 16:11	1553064	YES	NO	9	NONE
GEXT_DOCUMENT_DETAIL	DOCUMENT_VERSION_ID	2	C102	C10A	0.50000000	0	1	5/8/2009 16:11	5420	YES	NO	3	NONE
GEXT_DOCUMENT_MASTER	DOCUMENT_FILE_NAME	230432	3030303032352E70646	687474703A2F2F67707 36C6270646D30312E63 6F72706F726174652E6 7652E63	0.00000434	32	1	5/8/2009 16:11	57879	YES	NO	38	NONE
GEXT_DOCUMENT_MASTER	DOCUMENT_ID	231532	302D3130	74657374	0.00000432	0	1	5/8/2009 16:11	57887	YES	NO	16	NONE
GEXT_DOCUMENT_MASTER	DOCUMENT_ISSUED_DATE	6655	66640314010101	786B0C1F010101	0.00015026	174080	1	5/8/2009 16:11	14367	YES	NO	3	NONE
GEXT_DOCUMENT_MASTER	DOCUMENT_TITLE	115173	2B33352046205550524 15445	6E65772023322062656 172696E672042727573 6820536561 6C7320616 E64206E	0.00000868	98977	1	5/8/2009 16:11	133230	YES	NO	9	NONE
GEXT_DOCUMENT_MASTER	DOCUMENT_TYPE_ID	23	C210	C21022	0.04347826	0	1	5/8/2009 16:11	6889	YES	NO	4	NONE
GEXT_DOCUMENT_MASTER	DOCUMENT_VERSION_ID	1	C102	C102	1.00000000	0	1	5/8/2009 16:11	6889	YES	NO	3	NONE
GEXT_DOCUMENT_MASTER	PUBLISHER_ID	8	3E6466	C10C	0.12500000	33	1	5/8/2009 16:11	6888	YES	NO	4	NONE
GEXT_DOCUMENT_UNIT	DOCUMENT_ID	96849	303030303031	57393336303351	0.00001033	0	1	5/8/2009 16:11	41294	YES	NO	10	NONE
GEXT_DOCUMENT_UNIT	UNIT_SERIAL_NUMBER	3162	303834373847	554E4B4E574E	0.00031626	0	1	5/8/2009 16:11	4605	YES	NO	7	NONE
GEXT_FRAME_MASTER	FRAME_ID	15	3E6466	C20A0A	0.06666667	0	1	5/8/2009 16:11	15	YES	NO	4	NONE
GEXT_ITEM_CRITERIA	CAPABILITY_ITEM_ID	227	C21229	C23448	0.00440529	0	1	5/8/2009 16:11	453	YES	NO	4	NONE
GEXT_ITEM_CRITERIA	CRITERIA_ID	226	C104	C2135F	0.00442478	0	1	5/8/2009 16:11	453	YES	NO	4	NONE
GEXT_ITEM_MASTER	CAPITAL_PART	2	80	C102	0.50000000	268499	1	5/8/2009 16:05	3625	YES	NO	2	NONE
GEXT_ITEM_MASTER	DRAWING_NUMBER	1836	20202020202020202020 20202020202020202020	5A534C2031363141	0.00054466	0	1	5/8/2009 16:05	4251	YES	NO	20	NONE
GEXT_ITEM_MASTER	ENGINEERING_STATUS	1	55534541424C45	55534541424C45	1.00000000	1820478	1	5/8/2009 16:05	348	YES	NO	2	NONE
GEXT_ITEM_MASTER	ITEM_DESC	18380	20326E6420537461676 5204E6F7A7A6C65204 B697420202020	73694320466C616D652 04465746563746F7220 4B69742C2034	0.00005441	195842	1	5/8/2009 16:05	38226	YES	NO	23	NONE
GEXT_ITEM_MASTER	ITEM_NUMBER	1823308	28444D2D47523037393 129	5A59474C4F205A4C2D 363641	0.00000055	0	1	5/8/2009 16:05	4251	YES	NO	16	NONE
GEXT_ITEM_MASTER	MLI_ID	552	2D2D2D2D	55423238	0.00181159	0	1	5/8/2009 16:05	4251	YES	NO	5	NONE
GEXT_ITEM_MASTER	SUPERSEDED_BY	100	20202020202020202020 20202020202020202020	53502D353533372D373 5	0.01000000	359000	1	5/8/2009 16:05	3414	YES	NO	17	NONE
GEXT_ITEM_TYPE	ITEM_TYPE_ID	12	C202	C20E	0.08333333	0	1	5/8/2009 16:12	12	YES	NO	3	NONE
GEXT_MLI_MASTER	MLI_FILTER_IND	2	80	C102	0.50000000	182	1	5/8/2009 16:12	7965	YES	NO	2	NONE
GEXT_MLI_MASTER	MLI_ID	8147	00000000	AD	0.00012274	0	1	5/8/2009 16:12	8147	YES	NO	5	NONE
GEXT_PART_TYPE	ITEM_NUMBER	19629	31303144323038395030 3031	54455354	0.00005095	0	1	5/8/2009 16:12	20466	YES	NO	14	NONE
GEXT_PART_TYPE	PART_TYPE_DEF_ID	1247	C23302	C24338	0.00080192	0	1	5/8/2009 16:12	20466	YES	NO	4	NONE

Tabla 3.3 Estadísticas en columnas del esquema TBGEX de la vista USER_TAB_COLUMNS.

TABLE_NAME	COLUMN_NAME	NUM_DISTINCT	LOW_VALUE	HIGH_VALUE	DENSITY	NUM_NULLS	NUM_BUCKETS	LAST_ANALYZED	SAMPLE_SIZE	GLOBAL_STATS	USER_STATS	AVG_COL_LEN	HISTOGRAM
GEXT_PART_TYPE_DEFINITION	FRAME_ID	12	3E6466	C20A0A	0.08333333	0	1	5/8/2009 16:12	1523	YES	NO	4	NONE
GEXT_PART_TYPE_DEFINITION	ITEM_TYPE_ID	10	C202	C20E	0.10000000	0	1	5/8/2009 16:12	1523	YES	NO	3	NONE
GEXT_PART_TYPE_DEFINITION	PART_TYPE_DEF_ID	1523	3E6466	C24348	0.00065660	0	1	5/8/2009 16:12	1523	YES	NO	4	NONE
GEXT_PART_TYPE_DEFINITION	SYSTEM_DEFINITION_ID	150	3E6466	C22A3C	0.00666667	0	1	5/8/2009 16:12	1523	YES	NO	4	NONE
GEXT_PART_TYPE_DEFINITION	UNIT_TYPE_ID	2	3E6466	C102	0.50000000	0	1	5/8/2009 16:12	1523	YES	NO	4	NONE
GEXT_PERFORMANCE_DETAIL	FUEL_FOLDER_ID	4	C102	C164	0.25000000	0	1	5/8/2009 16:12	5467	YES	NO	3	NONE
GEXT_PERFORMANCE_DETAIL	IPS_DESIGN_NUMBER	21	4531	4739	0.04761905	0	1	5/8/2009 16:12	5467	YES	NO	4	NONE
GEXT_PERFORMANCE_DETAIL	IPS_NUMBER	3322	303030303239	533035303538	0.00030102	0	1	5/8/2009 16:12	5467	YES	NO	7	NONE
GEXT_PERFORMANCE_DETAIL	UNIT_SERIAL_NUMBER	6121	3130303733	54484D30303331	0.00016337	0	1	5/8/2009 16:12	5467	YES	NO	7	NONE
GEXT_PERFORMANCE_DOCUMENT	DOCUMENT_ID	3690	20	524D31373738	0.00027100	0	1	5/8/2009 16:12	7057	YES	NO	8	NONE
GEXT_PERFORMANCE_DOCUMENT	DOCUMENT_VERSION_ID	1	C102	C102	1.00000000	0	1	5/8/2009 16:12	7057	YES	NO	3	NONE
GEXT_PERFORMANCE_DOCUMENT	IPS_DESIGN_NUMBER	23	4531	4739	0.04347826	0	1	5/8/2009 16:12	7057	YES	NO	4	NONE
GEXT_PERFORMANCE_DOCUMENT	IPS_NUMBER	3363	303030303037	524D31373736	0.00029735	0	1	5/8/2009 16:12	7057	YES	NO	7	NONE
GEXT_PERFORMANCE_DOCUMENT	UNIT_SERIAL_NUMBER	5676	3130303733	54393638	0.00017618	0	1	5/8/2009 16:12	7057	YES	NO	7	NONE
GEXT_PERFORMANCE_HEADER	IPS_DESIGN_NUMBER	24	4531	4739	0.04166667	0	1	5/8/2009 16:12	8523	YES	NO	4	NONE
GEXT_PERFORMANCE_HEADER	IPS_NUMBER	4538	303030303037	554E4B4E4F574E	0.00022036	0	1	5/8/2009 16:12	8523	YES	NO	7	NONE
GEXT_PERFORMANCE_HEADER	UNIT_SERIAL_NUMBER	6822	3130303733	54484D30303331	0.00014658	0	1	5/8/2009 16:12	8523	YES	NO	7	NONE
GEXT_STAGE_MASTER	STAGE_ID	45	3E6466	C21F2D	0.02222222	0	1	5/8/2009 16:12	45	YES	NO	4	NONE
GEXT_SYSTEM_DEFINITION	COMPONENT_ID	29	3E6466	C2151D	0.03448276	0	1	5/8/2009 16:12	153	YES	NO	4	NONE
GEXT_SYSTEM_DEFINITION	STAGE_ID	45	3E6466	C21F2D	0.02222222	0	1	5/8/2009 16:12	153	YES	NO	4	NONE
GEXT_SYSTEM_DEFINITION	SYSTEM_DEFINITION_ID	153	3E6466	C22A3C	0.00653595	0	1	5/8/2009 16:12	153	YES	NO	4	NONE
GEXT_SYSTEM_DEFINITION	SYSTEM_ID	6	3E6466	C20B06	0.16666667	0	1	5/8/2009 16:12	153	YES	NO	4	NONE
GEXT_SYSTEM_MASTER	SYSTEM_ID	6	3E6466	C20B06	0.16666667	0	1	5/8/2009 16:12	6	YES	NO	4	NONE
GEXT_SYSTEM_TYPE_DEFINITION	ITEM_TYPE_ID	11	C202	C20E	0.09090909	0	1	5/8/2009 16:12	406	YES	NO	3	NONE
GEXT_SYSTEM_TYPE_DEFINITION	SYSTEM_DEFINITION_ID	150	3E6466	C22A3C	0.00666667	0	1	5/8/2009 16:12	406	YES	NO	4	NONE
GEXT_URS	AUTHOR_NM	315	323034303034343436	79757365	0.00317460	69978	1	5/8/2009 16:13	2885	YES	NO	5	NONE
GEXT_URS	AUTH_EC_NUMBER	15345	23303837 4E505536303 0303539	7370726616473686565 742031	0.00006517	0	1	5/8/2009 16:13	39130	YES	NO	13	NONE
GEXT_URS	ENTRY_DATE	3081	64650101010101	786B0418010101	0.00032457	0	1	5/8/2009 16:13	3903	YES	NO	8	NONE
GEXT_URS	MLI_ID	500	26262827	5A303132	0.00200000	0	1	5/8/2009 16:13	3903	YES	NO	5	NONE
GEXT_URS	NEW_ITEM_NUMBER	26726	2853454520523035303 43629	54494C2D313337352D3 2	0.00003742	0	1	5/8/2009 16:13	39130	YES	NO	13	NONE
GEXT_URS	OLD_ITEM_NUMBER	449	31	5343484544554C4544	0.00222717	0	1	5/8/2009 16:13	3903	YES	NO	6	NONE
GEXT_URS	QUANTITY	29	80	C21E	0.03448276	0	1	5/8/2009 16:13	3903	YES	NO	3	NONE
GEXT_URS	UNIT_SERIAL_NUMBER	2933	3130383737	54393638	0.00034095	0	1	5/8/2009 16:13	3903	YES	NO	7	NONE

Tabla 3.3 Estadísticas en columnas del esquema TBGEX de la vista USER_TAB_COLUMNS.

3.1.3.3 Estadísticas de índices.

TABLE_NAME	INDEX_NAME	BLEVEL	LEAF_BLOCKS	DISTINCT_KEYS	AVG_LEAF_BLO CKS_PER_KEY	AVG_DATA_BLO CKS_PER_KEY	CLUSTERING_FA CTOR	LAST_ANALYZED	SAMPLE_SIZE	USER_STATS	GLOBAL_STATS
GEXT_CAPABILITY_DETAIL	GEXI_CAPABILITY_DETAIL_DOC	1	62	608	1	1	551	5/8/2009 16:08	22420	NO	YES
GEXT_CAPABILITY_DETAIL	GEXI_CAPABILITY_DETAIL_PK	1	120	22420	1	1	517	5/8/2009 16:08	22420	NO	YES
GEXT_CAPABILITY_DETAIL	GEXI_CAPABILITY_DETL_DATE_AV	0	1	1	1	33	33	5/8/2009 16:08	37	NO	YES
GEXT_CAPABILITY_DETAIL	GEXI_CAPABILITY_DETL_NUM_AV	1	20	477	1	9	4511	5/8/2009 16:08	7746	NO	YES
GEXT_CAPABILITY_DETAIL	GEXI_CAPABILITY_DETL_UPPER_AV	1	70	549	1	9	5422	5/8/2009 16:08	19438	NO	YES
GEXT_CAPABILITY_ITEM_TYPE	GEXI_CAPABILITY_ITEM_TYPE_PK	1	3	1701	1	1	14	5/8/2009 16:08	1701	NO	YES
GEXT_CAPABILITY_MASTER	GEXI_CAPABILITY_MASTER_PK	1	2	608	1	1	382	5/8/2009 16:08	608	NO	YES
GEXT_CAPABILITY_MASTER	GEXI_CAPMASTER_CAPDOCVER	1	3	608	1	1	382	5/8/2009 16:08	608	NO	YES
GEXT_CAPABILITY_MASTER	GEXI_CAPMASTER_PCRITERIA	1	2	189	1	1	47	5/8/2009 16:08	608	NO	YES
GEXT_CAPABILITY_MASTER	GEXI_CAPMASTER_UCRITERIA	1	2	33	1	1	16	5/8/2009 16:08	608	NO	YES
GEXT_COMPONENT_MASTER	GEXI_COMPONENT_MASTER_PK	0	1	29	1	1	1	5/8/2009 16:08	29	NO	YES
GEXT_CONFIGURATION	GEXI_CONFIGURATION_IDX01	2	105508	3255862	1	9	32376672	5/8/2009 16:07	338208	NO	YES
GEXT_CONFIGURATION	GEXI_CONFIGURATION_IDX02	2	74658	10242	7	128	1314640	5/8/2009 16:07	5612600	NO	YES
GEXT_CONFIGURATION	GEXI_CONFIGURATION_IDX03	2	70662	10140	6	1101	11166528	5/8/2009 16:07	5330200	NO	YES
GEXT_CONFIGURATION	GEXI_CONFIGURATION_IF49	2	88313	4482	19	7407	33199600	5/8/2009 16:07	460884	NO	YES
GEXT_CONFIGURATION	GEXI_CONFIGURATION_IF54	2	104426	532444	1	46	24530177	5/8/2009 16:07	360922	NO	YES
GEXT_CONFIGURATION	GEXI_CONFIGURATION_IF83	3	164367	33785679	1	1	34844896	5/8/2009 16:07	253806	NO	YES
GEXT_CONFIGURATION	GEXI_CONFIGURATION_IF97	2	76553	766	99	18154	13906714	5/8/2009 16:08	504675	NO	YES
GEXT_CONFIGURATION	GEXI_CONFIGURATION_PK	2	104646	35736921	1	1	35323577	5/8/2009 16:07	402289	NO	YES
GEXT_CRITERIA_CHECKS	GEXI_CRITERIA_CHECKS_PK	1	3	974	1	1	43	5/8/2009 16:08	974	NO	YES
GEXT_CRITERIA_MASTER	GEXI_CRITERIA_MASTER_PK	0	1	259	1	1	3	5/8/2009 16:08	259	NO	YES
GEXT_DOCUMENT_DETAIL	GEXI_DOCUMENT_DETAIL1_PK	2	5600	1541966	1	1	10172	5/8/2009 16:11	313344	NO	YES
GEXT_DOCUMENT_DETAIL	GEXI_DOCUMENT_DETAIL_DATE_AV	1	140	10096	1	2	27546	5/8/2009 16:11	52497	NO	YES
GEXT_DOCUMENT_DETAIL	GEXI_DOCUMENT_DETAIL_IF63	2	3893	129777	1	1	5767	5/8/2009 16:11	1553064	NO	YES
GEXT_DOCUMENT_DETAIL	GEXI_DOCUMENT_DETAIL_IF64	2	5925	75	79	1102	82687	5/8/2009 16:11	322701	NO	YES
GEXT_DOCUMENT_DETAIL	GEXI_DOCUMENT_DETAIL_NUM_AV	1	392	18616	1	3	72453	5/8/2009 16:11	182582	NO	YES
GEXT_DOCUMENT_DETAIL	GEXI_DOCUMENT_DETAIL_UPPER_AV	2	3889	250693	1	1	351434	5/8/2009 16:11	1475076	NO	YES
GEXT_DOCUMENT_MASTER	GEXI_DOCUMENT_MASTER_IF78	2	755	14	53	289	4058	5/8/2009 16:11	232107	NO	YES
GEXT_DOCUMENT_MASTER	GEXI_DOCUMENT_MASTER_IF79	2	789	31	25	143	4453	5/8/2009 16:11	232205	NO	YES
GEXT_DOCUMENT_MASTER	GEXI_DOCUMENT_MASTER_PK	2	1239	232207	1	1	29827	5/8/2009 16:11	232207	NO	YES
GEXT_DOCUMENT_UNIT	GEXI_DOCUMENT_UNIT_IF121	2	585	7821	1	9	72477	5/8/2009 16:11	165268	NO	YES
GEXT_DOCUMENT_UNIT	GEXI_DOCUMENT_UNIT_IF122	1	419	96537	1	1	484	5/8/2009 16:11	165268	NO	YES
GEXT_DOCUMENT_UNIT	GEXI_DOCUMENT_UNIT_PK	2	604	165268	1	1	1069	5/8/2009 16:11	165268	NO	YES
GEXT_FRAME_MASTER	GEXI_FRAME_MASTER_PK	0	1	15	1	1	1	5/8/2009 16:11	15	NO	YES
GEXT_ITEM_CRITERIA	GEXI_ITEM_CRITERIA_PK	1	2	453	1	1	136	5/8/2009 16:11	453	NO	YES
GEXT_ITEM_MASTER	GEXI_ITEM_MASTER_IDX2	1	39	2	19	764	1529	5/8/2009 16:05	14196	NO	YES
GEXT_ITEM_MASTER	GEXI_ITEM_MASTER_IF61	2	9444	192626	1	1	232413	5/8/2009 16:05	213964	NO	YES
GEXT_ITEM_MASTER	GEXI_ITEM_MASTER_IF70	2	4140	1196	3	46	55657	5/8/2009 16:05	1846445	NO	YES
GEXT_ITEM_MASTER	GEXI_ITEM_MASTER_PK	2	6379	1820283	1	1	1803286	5/8/2009 16:05	322146	NO	YES
GEXT_ITEM_MASTER	GEXI_ITEM_MASTER_SUPERCEDED_BY	2	6904	100	69	713	71340	5/8/2009 16:05	287950	NO	YES
GEXT_ITEM_TYPE	GEXI_ITEM_TYPE_PK	0	1	12	1	1	1	5/8/2009 16:12	12	NO	YES
GEXT_MLI_MASTER	GEXI_MLI_MASTER_PK	1	28	8147	1	1	7825	5/8/2009 16:12	8147	NO	YES
GEXT_PART_TYPE	GEXI_PART_TYPE_IF75	1	67	1247	1	1	793	5/8/2009 16:12	20466	NO	YES
GEXT_PART_TYPE	GEXI_PART_TYPE_IF76	1	89	19629	1	1	6499	5/8/2009 16:12	20466	NO	YES
GEXT_PART_TYPE	GEXI_PART_TYPE_PK	1	78	20466	1	1	1088	5/8/2009 16:12	20466	NO	YES
GEXT_PART_TYPE_DEFINITION	GEXI_PART_TYPE_DEFINITION_IF73	1	5	12	1	3	36	5/8/2009 16:12	1523	NO	YES
GEXT_PART_TYPE_DEFINITION	GEXI_PART_TYPE_DEFINITION_IF74	1	5	403	1	1	660	5/8/2009 16:12	1523	NO	YES
GEXT_PART_TYPE_DEFINITION	GEXI_PART_TYPE_DEFINITION_PK	1	4	1523	1	1	752	5/8/2009 16:12	1523	NO	YES
GEXT_PERFORMANCE_DETAIL	GEXI_PERFORMANCE_DETAIL_IF31	2	1584	89	17	1825	162454	5/8/2009 16:12	533569	NO	YES
GEXT_PERFORMANCE_DETAIL	GEXI_PERFORMANCE_DETAIL_IF41	2	3000	8501	1	1	15097	5/8/2009 16:12	533569	NO	YES
GEXT_PERFORMANCE_DETAIL	GEXI_PERFORMANCE_DETAIL_IF58	2	1558	4	389	2063	8255	5/8/2009 16:12	533569	NO	YES
GEXT_PERFORMANCE_DETAIL	GEXI_PERFORMANCE_DETAIL_PK	2	3735	533569	1	1	39453	5/8/2009 16:12	533569	NO	YES
GEXT_PERFORMANCE_DETAIL	GEXI_PERFORMANCE_DETL_DATE_AV	1	26	1351	1	4	6498	5/8/2009 16:12	9500	NO	YES
GEXT_PERFORMANCE_DETAIL	GEXI_PERFORMANCE_DETL_NUM_AV	1	289	16196	1	5	82167	5/8/2009 16:12	140236	NO	YES
GEXT_PERFORMANCE_DETAIL	GEXI_PERFORMANCE_DETL_UPPER_AV	2	1355	40771	1	4	177610	5/8/2009 16:12	507616	NO	YES
GEXT_PERFORMANCE_DOCUMENT	GEXI_PERFORMANCE_DOCUMENT_IF72	1	26	3690	1	1	2094	5/8/2009 16:12	7057	NO	YES
GEXT_PERFORMANCE_DOCUMENT	GEXI_PERFORMANCE_DOCUMENT_PK	1	36	7057	1	1	2967	5/8/2009 16:12	7057	NO	YES
GEXT_PERFORMANCE_HEADER	GEXI_PERFORMANCE_HEADER_IF43	1	28	6822	1	1	4089	5/8/2009 16:12	8523	NO	YES
GEXT_PERFORMANCE_HEADER	GEXI_PERFORMANCE_HEADER_PK	1	32	8523	1	1	4092	5/8/2009 16:12	8523	NO	YES
GEXT_STAGE_MASTER	GEXI_STAGE_MASTER_PK	0	1	45	1	1	1	5/8/2009 16:12	45	NO	YES
GEXT_SYSTEM_DEFINITION	GEXI_SYSTEM_DEFINITION_AK1	0	1	153	1	1	1	5/8/2009 16:12	153	NO	YES
GEXT_SYSTEM_DEFINITION	GEXI_SYSTEM_DEFINITION_IF80	0	1	6	1	1	1	5/8/2009 16:12	153	NO	YES
GEXT_SYSTEM_DEFINITION	GEXI_SYSTEM_DEFINITION_IF81	0	1	29	1	1	1	5/8/2009 16:12	153	NO	YES
GEXT_SYSTEM_DEFINITION	GEXI_SYSTEM_DEFINITION_IF82	0	1	45	1	1	1	5/8/2009 16:12	153	NO	YES
GEXT_SYSTEM_DEFINITION	GEXI_SYSTEM_DEFINITION_PK	0	1	153	1	1	1	5/8/2009 16:12	153	NO	YES
GEXT_SYSTEM_MASTER	GEXI_SYSTEM_MASTER_PK	0	1	6	1	1	1	5/8/2009 16:12	6	NO	YES
GEXT_SYSTEM_TYPE_DEFINITION	GEXI_SYSTEM_TYPE_DEFINITION_PK	0	1	406	1	1	1	5/8/2009 16:12	406	NO	YES
GEXT_SYSTEM_TYPE_DEFINITION	GEXI_SYSTEM_TYPE_DEFINITI_IF49	0	1	150	1	1	1	5/8/2009 16:12	406	NO	YES
GEXT_SYSTEM_TYPE_DEFINITION	GEXI_SYSTEM_TYPE_DEFINITI_IF50	0	1	11	1	1	1	5/8/2009 16:12	406	NO	YES
GEXT_URS	GEXI_URS_IF100	2	1153	72093	1	3	224134	5/8/2009 16:13	269743	NO	YES
GEXT_URS	GEXI_URS_IF119	2	801	1506	1	105	158768	5/8/2009 16:13	269743	NO	YES
GEXT_URS	GEXI_URS_IF68	2	1117	5688	1	3	21704	5/8/2009 16:13	269743	NO	YES
GEXT_URS	GEXI_URS_IF99	2	970	8429	1	4	36555	5/8/2009 16:13	269743	NO	YES
GEXT_URS	GEXI_URS_PK	2	1967	269743	1	1	37926	5/8/2009 16:13	269743	NO	YES

Tabla 3.4 Estadísticas en índices del esquema TBGEX de la vista USER_INDEXES.

La columna LAST_ANALYZED muestra que las estadísticas se generaron recientemente. Las estadísticas de este esquema se colectan diariamente mediante la tarea GATHER_STATS_JOB.

3.1.4 Diseño del caso práctico.

3.1.4.1 Privilegios de base de datos requeridos.

En primera instancia se solicitaron al administrador de la base de datos los siguientes privilegios:

- SELECT_CATALOG_ROLE, para tener privilegios de lectura en todas las vistas del diccionario de datos.
- Rol PLUSTRACE, para la ejecución de la herramienta AUTOTRACE dentro de SQL*Plus.
- Privilegio ADVISOR, para la utilización de los procedimientos del paquete DBMS_SQLTUNE.

3.1.4.2 Scripts de SQL*Plus generados.

Utilizando un total de veintidós consultas sobre el esquema TBGEX, se crearon los siguientes scripts:

- optimizer_session.sql Es el archivo principal de ejecución. Contiene instrucciones para establecer diferentes variables de ambiente dentro de la sesión de SQL*Plus. Invoca a la herramienta AUTOTRACE para rastrear la ejecución de las consultas al emplear ya sea el enfoque de optimización basado en costos o basado en reglas.

```
SPOOL "C:\Documents and Settings\santibka\My Documents\Tesis\Output\optimizer_session_output.txt"

SET SQLPROMPT "_USER'@'_CONNECT_IDENTIFIER> "

SET ECHO ON
--Controla el despliegue de los comandos en la pantalla.

SET TIMING ON
--Para medir el tiempo de una consulta en Oracle SQL*Plus
--Controla el despliegue de estadísticas de tiempo
--en cada comando SQL o bloque PL/SQL ejecutado.

SET LINESIZE 180
```

```
--Establece el número total de caracteres que SQL*Plus
--despliega en una línea antes de comenzar una línea nueva.

SET TRIMSPOOL ON
--Determina si SQL*Plus pone blancos al final de cada línea enviada al archivo

SELECT TO_CHAR(SYSDATE, 'DD-MON-RR HH.MI.SS AM') TODAY_DATE FROM DUAL;

SELECT * FROM SESSION_ROLES;
--Rol PLUSTRACE

SET AUTOTRACE TRACEONLY
--Comando AUTOTRACE

SPOOL OFF;

--@"C:\Documents and Settings\santibka\My Documents\Tesis\rbo_session.sql"
--@"C:\Documents and Settings\santibka\My Documents\Tesis\cbo_session.sql"

EXIT
```

- rbo_session.sql Establece un enfoque de optimización basado en reglas para todas las sentencias.

```
SPOOL "C:\Documents and Settings\santibka\My Documents\Tesis\Output\sql_queries_rbo_output.txt"

ALTER SESSION SET OPTIMIZER_MODE = RULE;
--Establece un enfoque de optimización basado en reglas para todas las sentencias.

@"C:\Documents and Settings\santibka\My Documents\Tesis\sql_queries.sql"

SPOOL OFF;
```

- cbo_session.sql Establece un enfoque de optimización basado en costos para todas las sentencias y optimiza con el objetivo de mejor procesamiento.

```
SPOOL "C:\Documents and Settings\santibka\My Documents\Tesis\Output\sql_queries_cbo_output.txt"

ALTER SESSION SET OPTIMIZER_MODE = ALL_ROWS;
--Establece un enfoque de optimización basado en costos para todas las sentencias.

@"C:\Documents and Settings\santibka\My Documents\Tesis\sql_queries.sql"

SPOOL OFF;
```

- sql_queries.sql Contiene las veintidós consultas a ejecutar. Las consultas se caracterizan por ser equirreuniones y contener literales.

```
--GEXP_EDO_GET_ASSOCCAPAB_ENG
SELECT DISTINCT /* TAG 1 */ CM.CAPABILITY_ID
          FROM GEXT_CAPABILITY_MASTER CM, GEXT_DOCUMENT_MASTER DM
```

```
          WHERE (    (CM.DOCUMENT_ID = DM.DOCUMENT_ID)
                 AND (CM.DOCUMENT_VERSION_ID = DM.DOCUMENT_VERSION_ID)
                 AND ((CM.PART_CRITERIA_ID = 1301) OR (CM.UNIT_CRITERIA_ID = 1301)));

--GEXP_EDO_GET_BOMITEMLIST_COM
SELECT /* TAG 2 */ *
  FROM GEXV_CONFIGURATION CFG
 WHERE ((CFG.EQUIPMENT = '282625') AND (CFG.PARENT_BMFL_SEQUENCE = 1148));

--GEXP_EDO_GET_CAPABDATAFLDR_ENG
SELECT DISTINCT /* TAG 3 */ CAPD.ATTRIBUTE_ID, CAPD.ATTRIBUTE_VALUE
           FROM GEXT_CAPABILITY_DETAIL CAPD, GEXT_CAPABILITY_MASTER CAPM
          WHERE (    (CAPM.DOCUMENT_ID = '00-883.2')
                 AND (CAPM.DOCUMENT_VERSION_ID = 9)
                 AND (CAPM.CAPABILITY_ID = CAPD.CAPABILITY_ID)
                 AND (CAPM.DOCUMENT_VERSION_ID = CAPD.DOCUMENT_VERSION_ID)
                 AND (CAPD.CAPABILITY_ID = 1502)
                 AND (CAPD.FOLDER_ID = 101));

--GEXP_EDO_GET_CRITERIA_VAL
SELECT /* TAG 4 */ CRITERIA_ABBR VALUE_SHORT, CRITERIA_NAME VALUE_MEDIUM,
        CRITERIA_LONG_NAME VALUE_LONG, CRITERIA_DESCRIPTION VALUE_FULL,
        CRITERIA_SORT_ORDER FORMAT_DISPLAY_ORDER, CRITERIA_DISP_IND FORMAT_DISPLAY_IND
  FROM GEXT_CRITERIA_MASTER
 WHERE ((CRITERIA_ID = 1310));

--GEXP_EDO_GET_DOCCAPABDATA_ENG
SELECT DISTINCT /* TAG 5 */ CAPM.CAPABILITY_ID
           FROM GEXT_CAPABILITY_MASTER CAPM, GEXT_DOCUMENT_MASTER DM
          WHERE (    (CAPM.DOCUMENT_ID = DM.DOCUMENT_ID)
                 AND (CAPM.DOCUMENT_VERSION_ID = DM.DOCUMENT_VERSION_ID)
                 AND (CAPM.DOCUMENT_ID = '00-883.2'));

--GEXP_EDO_GET_DOCDATAATTRB_VAL
SELECT DISTINCT /* TAG 6 */ DD.ATTRIBUTE_VALUE
           FROM GEXT_DOCUMENT_DETAIL DD, GEXT_DOCUMENT_MASTER DM
          WHERE (    (DD.DOCUMENT_ID = DM.DOCUMENT_ID)
                 AND (DD.DOCUMENT_VERSION_ID = DM.DOCUMENT_VERSION_ID)
                 AND (DM.DOCUMENT_ID = 'GR0125')
                 AND (DD.ATTRIBUTE_ID = 1516));

--GEXP_EDO_GET_DOCPERFDATA_ENG
SELECT DISTINCT /* TAG 7 */ PDOC.UNIT_SERIAL_NUMBER
           FROM GEXT_PERFORMANCE_DOCUMENT PDOC, GEXT_DOCUMENT_MASTER DM
          WHERE (    (PDOC.DOCUMENT_ID = DM.DOCUMENT_ID)
                 AND (PDOC.DOCUMENT_VERSION_ID = DM.DOCUMENT_VERSION_ID)
                 AND (DM.DOCUMENT_ID = 'GR0125'));

--GEXP_EDO_GET_DOCUMENT_ENG
SELECT DISTINCT /* TAG 8 */ DM.DOCUMENT_FILE_NAME, DM.DOCUMENT_TITLE, DM.DOCUMENT_ISSUED_DATE,
                DM.PUBLISHER_ID, DM.DOCUMENT_TYPE_ID
           FROM GEXT_DOCUMENT_MASTER DM
          WHERE ((DM.DOCUMENT_ID = 'GR0125'));

--GEXP_EDO_GET_EQPBOMMANUMLI_COM
SELECT DISTINCT /* TAG 9 */ CFG.*
           FROM GEXV_CONFIGURATION CFG, GEXT_MLI_MASTER MLI
          WHERE (    (MLI.MLI_ID = CFG.MLITYPE)
```

```
                AND (CFG.EQUIPMENT = '282625')
                AND (MLI.MLI_FILTER_IND = 1));

--GEXP_EDO_GET_EQPBOMMRGCAP_COM
SELECT DISTINCT /* TAG 10 */ URS.*
          FROM GEXV_URS URS, GEXT_ITEM_MASTER IM
         WHERE (    (IM.ITEM_NUMBER = URS.PART)
                AND (URS.EQUIPMENT = '282625')
                AND (IM.CAPITAL_PART = 1));

--GEXP_EDO_GET_EQPBOMMRGMLI_COM
SELECT DISTINCT /* TAG 11 */ URS.*
          FROM GEXV_URS URS, GEXT_MLI_MASTER MLI
         WHERE (    (MLI.MLI_ID = URS.MLITYPE)
                AND (URS.EQUIPMENT = '282625')
                AND (MLI.MLI_FILTER_IND = 1));

--GEXP_EDO_GET_EQPMANFWPART_COM
SELECT /* TAG 12 */ CFG.UNIT_SERIAL_NUMBER EQUIPMENT, CFG.BMFL_SEQUENCE BMFLSEQNO,
       CFG.ITEM_NUMBER PART, CFG.MLI_ID MLITYPE, CFG.QUANTITY QUANTITY, 'EA' UNITOFMEASURE,
       CFG.UOM_NAME UNITOFMEASURE, CFG.BOM_LEVEL BOMLEVEL
  FROM GEXT_CONFIGURATION CFG
 WHERE ((CFG.ITEM_NUMBER = '109E3919G001'));

--GEXP_EDO_GET_EQUIPADDPART_COM
SELECT /* TAG 13 */ URS.*
  FROM GEXV_URS URS
 WHERE ((URS.PART = '109E3919G001'));

--GEXP_EDO_GET_EQUIPCRITERIA_ENG
SELECT DISTINCT /* TAG 14 */ CRD.CHECK_ID, CRD.SORT_ORDER
          FROM GEXT_CRITERIA_CHECKS CRD
         WHERE ((CRD.CRITERIA_ID = 1310))
      ORDER BY CRD.SORT_ORDER;

--GEXP_EDO_GET_EQUIPDOC_ENG
SELECT /* TAG 15 */ DOCUMENT_ID
  FROM GEXT_DOCUMENT_UNIT
 WHERE ((UNIT_SERIAL_NUMBER = '282625'));

--GEXP_EDO_GET_PART_ENG
SELECT DISTINCT /* TAG 16 */ IM.SUPERSEDED_BY, IM.ENGINEERING_STATUS, IM.ITEM_DESC,
                GEXF_GET_ENGINEERING_DESC ('109E3919G001') ENG_DESC, IM.DRAWING_NUMBER, IM.MLI_ID
          FROM GEXT_ITEM_MASTER IM
         WHERE ((IM.ITEM_NUMBER = '109E3919G001'));

--GEXP_EDO_GET_PART_ENG
SELECT DISTINCT /* TAG 17 */ NVL (PTP.PART_TYPE_DEF_ID, -1)
          FROM GEXT_ITEM_MASTER IM, GEXT_PART_TYPE PTP
         WHERE ((IM.ITEM_NUMBER = PTP.ITEM_NUMBER(+)) AND (IM.ITEM_NUMBER = '109E3919G001'));

--GEXP_EDO_GET_PARTCAPABDATA_ENG
SELECT /* TAG 18 */ ITC.CRITERIA_ID
  FROM GEXT_CAPABILITY_ITEM_TYPE CITP, GEXT_ITEM_CRITERIA ITC
 WHERE ((CITP.CAPABILITY_ITEM_ID = ITC.CAPABILITY_ITEM_ID) AND (CITP.ITEM_NUMBER = '354B3750G003'));

--GEXP_EDO_GET_PARTCRITERIA_ENG
SELECT DISTINCT /* TAG 19 */ CITP.ITEM_NUMBER
```

```
           FROM GEXT_CAPABILITY_ITEM_TYPE CITP, GEXT_ITEM_CRITERIA CICR
          WHERE (    (CITP.CAPABILITY_ITEM_ID = CICR.CAPABILITY_ITEM_ID)
                 AND (CICR.CRITERIA_ID = 1310)
                 AND (CITP.CAPABILITY_ITEM_TYPE = 'ITEM_NUMBER'));

--GEXP_EDO_GET_PARTTYPE_COM
SELECT DISTINCT /* TAG 20 */ PTD.UNIT_TYPE_ID, PTD.FRAME_ID, SD.SYSTEM_ID, SD.COMPONENT_ID,
                SD.STAGE_ID, PTD.ITEM_TYPE_ID
           FROM GEXT_STAGE_MASTER STGM,
                GEXT_COMPONENT_MASTER CM,
                GEXT_ITEM_TYPE ITP,
                GEXT_PART_TYPE_DEFINITION PTD,
                GEXT_SYSTEM_DEFINITION SD,
                GEXT_SYSTEM_MASTER SM,
                GEXT_SYSTEM_TYPE_DEFINITION STD,
                GEXT_FRAME_MASTER FM
          WHERE (    (STGM.STAGE_ID = SD.STAGE_ID)
                 AND (CM.COMPONENT_ID = SD.COMPONENT_ID)
                 AND (ITP.ITEM_TYPE_ID = STD.ITEM_TYPE_ID)
                 AND (STD.SYSTEM_DEFINITION_ID = PTD.SYSTEM_DEFINITION_ID)
                 AND (STD.ITEM_TYPE_ID = PTD.ITEM_TYPE_ID)
                 AND (SM.SYSTEM_ID = SD.SYSTEM_ID)
                 AND (SD.SYSTEM_DEFINITION_ID = STD.SYSTEM_DEFINITION_ID)
                 AND (FM.FRAME_ID = PTD.FRAME_ID)
                 AND (PTD.PART_TYPE_DEF_ID = 5775));

--GEXP_EDO_GET_PERFORMANCE_ENG
SELECT DISTINCT /* TAG 21 */ PDTL.FUEL_FOLDER_ID
           FROM GEXT_PERFORMANCE_DETAIL PDTL,
                GEXT_PERFORMANCE_DOCUMENT PDOC,
                GEXT_PERFORMANCE_HEADER PH
          WHERE (    (PH.UNIT_SERIAL_NUMBER = PDTL.UNIT_SERIAL_NUMBER)
                 AND (PH.IPS_NUMBER = PDTL.IPS_NUMBER)
                 AND (PH.IPS_DESIGN_NUMBER = PDTL.IPS_DESIGN_NUMBER)
                 AND (PH.UNIT_SERIAL_NUMBER = PDOC.UNIT_SERIAL_NUMBER)
                 AND (PH.IPS_NUMBER = PDOC.IPS_NUMBER)
                 AND (PH.IPS_DESIGN_NUMBER = PDOC.IPS_DESIGN_NUMBER)
                 AND (PH.UNIT_SERIAL_NUMBER = '282625')
                 AND (PDOC.DOCUMENT_ID = 'GR0169'));

--GEXP_EDO_GET_RELATEDUNITS_ENG
SELECT /* TAG 22 */ DU.UNIT_SERIAL_NUMBER
  FROM GEXT_DOCUMENT_UNIT DU
 WHERE ((DU.DOCUMENT_ID = 'GR0169'));
```

La secuencia de ejecución de estos archivos se muestra en la figura siguiente:

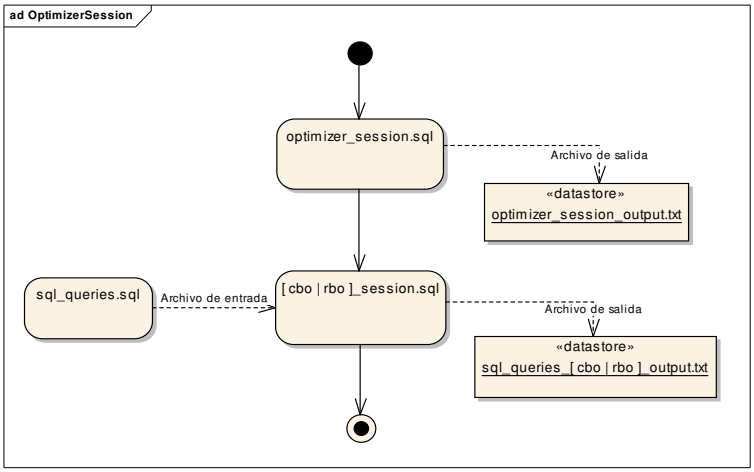

Figura 3.6 Diagrama de ejecución de los scripts generados para el caso práctico de optimización de consultas.

El script principal, optimizer_session.sql, puede invocar el enfoque de optimización basado en costos o basado en reglas llamando al archivo cbo_session.sql ó rbo_session.sql respectivamente.

La salida de los comandos ejecutados por el script optimizer_session.sql se guarda en el archivo optimizer_session_output.txt

Los archivos de cada enfoque de optimización, cbo_session.sql y rbo_session.sql, hacen uso del archivo sql_queries.sql que contiene las consultas a ejecutar.

Los resultados de ejecución generados por la herramienta AUTOTRACE de las consultas involucradas se guardan en los archivos de texto sql_queries_cbo_output.txt y sql_queries_rbo_output.txt según corresponda.

Las ejecuciones de las consultas bajo cada enfoque de optimización se llevaron a cabo en días diferentes y en un horario vespertino para tratar de asegurar que los datos involucrados no se encontraran en el búfer caché del SGA, lo anterior debido a que no se tienen privilegios de DBA para limpiar el búfer caché y el shared pool mediante el comando ALTER SYSTEM FLUSH BUFFER_CACHE | SHARED_POOL. [57]

3.1.4.3 Regeneración de estadísticas de la base de datos.

La tarea GATHER_STATS_JOB que incluye Oracle 10g reúne estadísticas en todos los objetos de la base de datos que tienen estadísticas faltantes o anticuadas. Esta tarea por defecto genera histogramas cuando reúne estadísticas de columna. Sin embargo para la ejecución del optimizador basado en costos se regeneraron las estadísticas del diccionario de datos de forma que no se generen histogramas y el optimizador conozca por columna únicamente el valor máximo, mínimo y el número de valores distintos.

Para eliminar y recrear las estadísticas de cada tabla se empleó el siguiente código PL/SQL.

```
BEGIN
  SYS.DBMS_STATS.DELETE_TABLE_STATS (
      OwnName         => 'TBGEX'
     ,TabName         => 'GEXT_DOCUMENT_MASTER'
    ,Cascade_Columns   => TRUE
    ,Cascade_Indexes   => TRUE
    ,Cascade_Parts     => TRUE
    ,No_Invalidate     => FALSE);
END;
/
```

Tabla 3.5 Código PL/SQL para eliminar las estadísticas de una tabla.

```
BEGIN
  SYS.DBMS_STATS.GATHER_TABLE_STATS (
      OwnName         => 'TBGEX'
     ,TabName         => 'GEXT_DOCUMENT_MASTER'
    ,Estimate_Percent  => SYS.DBMS_STATS.AUTO_SAMPLE_SIZE
    ,Method_Opt        => 'FOR ALL COLUMNS SIZE 1'
    ,Degree            => SYS.DBMS_STATS.DEFAULT_DEGREE
    ,Cascade           => TRUE
    ,No_Invalidate     => FALSE);
END;
/
```

Tabla 3.6 Código PL/SQL para reunir las estadísticas de una tabla sin histogramas.

4 RESULTADOS Y DISCUSIÓN

El resultado de la ejecución de cada consulta incluye:

1. Elapsed, periodo de tiempo transcurrido en la ejecución del comando.
2. Plan hash value, contiene la representación numérica del plan SQL.
3. Plan de ejecución de la consulta.
4. Estadísticas de SQL*Plus AUTOTRACE.

4.1 Estadísticas obtenidas mediante SQL*Plus AUTOTRACE.

Las estadísticas obtenidas mediante SQL*Plus AUTOTRACE para la sesión del optimizador basado en costos y basado en reglas se muestran en las tablas siguientes.

Query	Elapsed	Plan hash value	recursive calls	db block gets	consistent gets	physical reads	redo size	bytes sent via SQL*Net to client	bytes received via SQL*Net from client	SQL*Net roundtrips to/from client	sorts (memory)	sorts (disk)	rows processed
1	00:00:01.43	1541948779	1	0	61	9	0	422	246	3	0	0	20
2	00:00:01.39	2012301996	179	0	81	46	116	563	239	2	10	0	4
3	00:00:01.21	2612966111	15	0	9	4	0	354	239	2	0	0	6
4	00:00:01.18	2122101414	1	0	2	2	0	445	239	2	0	0	1
5	00:00:01.31	3508301604	1	0	9	3	0	464	246	3	0	0	24
6	00:00:01.23	2035648940	15	0	11	6	0	247	239	2	0	0	1
7	00:00:01.32	1049795864	1	0	9	3	0	260	239	2	0	0	2
8	00:00:01.17	719383829	1	0	4	1	0	428	239	2	0	0	1
9	00:00:35.82	4069526553	235	0	10100	9973	0	1570	253	4	3	0	34
10	00:00:02.28	3324069663	706	0	456	138	0	1267	239	2	21	0	14
11	00:00:01.46	642984172	88	0	102	0	0	2626	253	4	0	0	39
12	00:00:01.54	4258698081	1	0	32	28	0	1428	246	3	0	0	27
13	00:00:01.51	1548254942	8	0	25	21	0	1395	246	3	0	0	22
14	00:00:01.18	1095053854	1	0	3	3	0	282	239	2	1	0	2
15	00:00:01.76	2148511359	1	0	47	39	0	930	260	5	0	0	51
16	00:00:01.46	3185817551	117	0	64	23	0	547	239	2	13	0	1
17	00:00:01.18	316242083	1	0	6	3	0	246	239	2	0	0	1
18	00:00:01.34	1043000449	60	0	39	25	0	255	239	2	0	0	3
19	00:00:01.18	3770764186	15	0	12	3	0	239	239	2	0	0	1
20	00:00:01.26	3328739327	92	0	49	7	0	407	239	2	0	0	1
21	00:00:01.31	1040190692	15	0	13	7	0	248	239	2	0	0	2
22	00:00:01.15	4097211151	1	0	4	0	0	241	239	2	0	0	1

Tabla 4.1 Estadísticas de SQL*Plus AUTOTRACE - Optimizador Basado en Costos.

Query	Elapsed	Plan hash value	recursive calls	db block gets	consistent gets	physical reads	redo size	bytes sent via SQL*Net to client	bytes received via SQL*Net from client	SQL*Net roundtrips to/from client	sorts (memory)	sorts (disk)	rows processed
1	00:00:02.07	407829381	1423	0	474	13	0	422	246	3	21	0	20
2	00:00:01.54	2323000060	784	0	1227	25	0	565	239	2	15	0	4
3	00:00:01.51	311556766	464	0	176	25	0	356	239	2	5	0	6
4	00:00:01.34	2122101414	60	0	14	2	0	447	239	2	2	0	1
5	00:00:01.62	3374727904	1	0	62	8	0	467	246	3	1	0	24
6	00:00:05.87	4036376586	907	0	97406	2989	0	249	239	2	11	0	1
7	00:00:01.25	3957246259	640	0	192	3	0	262	239	2	11	0	2
8	00:00:01.28	1256613531	1	0	4	1	0	430	239	2	1	0	1
9	00:00:01.96	188307742	1239	0	609	99	0	1442	253	4	17	0	34
10	00:00:24.76	3889829912	2190	0	80016	31403	0	1236	239	2	40	0	14
11	00:00:01.96	3948118121	293	0	661	51	0	2570	253	4	5	0	39
12	00:00:01.81	4258698081	1	0	32	30	0	1433	246	3	0	0	27
13	00:00:01.79	1548254942	8	0	25	18	0	1398	246	3	0	0	22
14	00:00:01.32	1095053854	305	0	70	5	0	284	239	2	7	0	2
15	00:00:01.95	2148511359	410	0	150	40	0	935	260	5	6	0	51
16	00:00:11.01	313243777	462	0	31650	30879	0	549	239	2	11	0	1
17	00:00:08.28	3275503428	132	0	31505	30856	0	248	239	2	3	0	1
18	00:00:01.36	3105789856	612	0	1521	14	0	257	239	2	12	0	3
19	00:00:01.28	382497427	15	0	11	2	0	242	239	2	1	0	1
20	00:00:01.37	1960812001	3460	0	930	15	0	409	239	2	63	0	1
21	00:00:01.37	2656976093	999	0	304	13	0	250	239	2	13	0	2
22	00:00:01.20	4097211151	1	0	4	2	0	243	239	2	0	0	1

Tabla 4.2 Estadísticas de SQL*Plus AUTOTRACE - Optimizador Basado en Reglas.

En las tablas anteriores se observa que las consultas con etiqueta 4, 12, 13, 14, 15 y 22 emplearon el mismo plan de ejecución con el optimizador basado en costos y el optimizador basado en reglas, ya que sus valores en la estadística PLAN_HASH_VALUE coinciden.

A continuación se ilustran los planes de ejecución generados con el enfoque de optimización basado en costos y basado en reglas para la consulta con etiqueta 4.

```
TBGEX@atlord17> --GEXP_EDO_GET_CRITERIA_VAL
TBGEX@atlord17> SELECT /* TAG 4 */ CRITERIA_ABBR VALUE_SHORT, CRITERIA_NAME VALUE_MEDIUM,
  2        CRITERIA_LONG_NAME VALUE_LONG, CRITERIA_DESCRIPTION VALUE_FULL,
  3        CRITERIA_SORT_ORDER FORMAT_DISPLAY_ORDER,
  4        CRITERIA_DISP_IND FORMAT_DISPLAY_IND
  5    FROM GEXT_CRITERIA_MASTER
  6   WHERE ((CRITERIA_ID = 1310));

Elapsed: 00:00:01.18

Execution Plan
----------------------------------------------------------
Plan hash value: 2122101414

-------------------------------------------------------------------------------------------------
| Id  | Operation                   | Name                    | Rows  | Bytes | Cost (%CPU)| Time     |
-------------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                         |     1 |    69 |     1   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| GEXT_CRITERIA_MASTER    |     1 |    69 |     1   (0)| 00:00:01 |
|*  2 |   INDEX UNIQUE SCAN         | GEXI_CRITERIA_MASTER_PK |     1 |       |     0   (0)| 00:00:01 |
-------------------------------------------------------------------------------------------------

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

   2 - access("CRITERIA_ID"=1310)


Statistics
----------------------------------------------------------
         1  recursive calls
         0  db block gets
         2  consistent gets
         2  physical reads
         0  redo size
       445  bytes sent via SQL*Net to client
       239  bytes received via SQL*Net from client
         2  SQL*Net roundtrips to/from client
         0  sorts (memory)
         0  sorts (disk)
         1  rows processed
```

Figura 4.1 Plan de ejecución para la consulta con etiqueta 4 con el optimizador basado en costos.

```
TBGEX@atlord17> --GEXP_EDO_GET_CRITERIA_VAL
TBGEX@atlord17> SELECT /* TAG 4 */ CRITERIA_ABBR VALUE_SHORT, CRITERIA_NAME
VALUE_MEDIUM,
  2        CRITERIA_LONG_NAME VALUE_LONG, CRITERIA_DESCRIPTION VALUE_FULL,
  3        CRITERIA_SORT_ORDER FORMAT_DISPLAY_ORDER,
  4        CRITERIA_DISP_IND FORMAT_DISPLAY_IND
  5    FROM GEXT_CRITERIA_MASTER
  6   WHERE ((CRITERIA_ID = 1310));

Elapsed: 00:00:01.34

Execution Plan
----------------------------------------------------------
Plan hash value: 2122101414

--------------------------------------------------------------------------
| Id  | Operation                   | Name                    |
--------------------------------------------------------------------------
|   0 | SELECT STATEMENT            |                         |
|   1 |  TABLE ACCESS BY INDEX ROWID| GEXT_CRITERIA_MASTER    |
|*  2 |   INDEX UNIQUE SCAN         | GEXI_CRITERIA_MASTER_PK |
--------------------------------------------------------------------------

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

   2 - access("CRITERIA_ID"=1310)

Note
-----
   - rule based optimizer used (consider using cbo)


Statistics
----------------------------------------------------------
        60  recursive calls
         0  db block gets
        14  consistent gets
         2  physical reads
         0  redo size
       447  bytes sent via SQL*Net to client
       239  bytes received via SQL*Net from client
         2  SQL*Net roundtrips to/from client
         2  sorts (memory)
         0  sorts (disk)
         1  rows processed
```

Figura 4.2 Plan de ejecución para la consulta con etiqueta 4 con el optimizador basado en reglas.

4.1.1 Análisis de la estadística “consistent gets”.

La gráfica siguiente compara el número de bloques procesados en modo de lectura consistente por consulta, la estadística “consistent gets”, entre el optimizador basado en costos y basado en reglas.

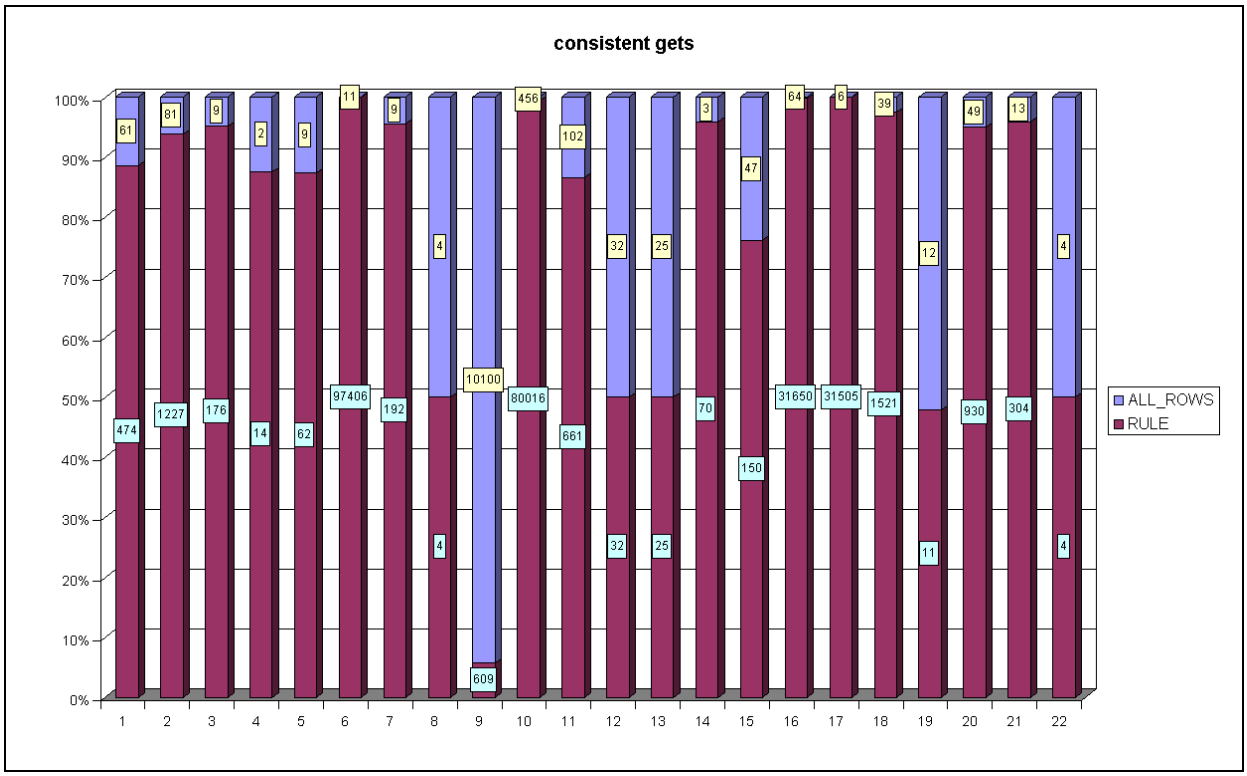

Figura 4.3 Comparación por consulta de la estadística “consistent gets”, entre el optimizador basado en costos y basado en reglas.

Con el optimizador basado en reglas, la consulta con etiqueta 2 realizó 1,227 lecturas consistentes, mientras que con el optimizador basado en costos llevó a cabo 81 lecturas. Las lecturas consistentes realizadas con el optimizador basado en reglas para la consulta con etiqueta 2 representan más del 90% del total de lecturas realizadas entre ambos optimizadores para esta consulta. Lo mismo se observa para las consultas con etiqueta 3, 6, 7, 10, 14, 16, 17, 18, 20 y 21.

En cambio con el optimizador basado en costos, sólo la consulta con etiqueta 9 realizó un total de lecturas consistentes que representa más del 90% del total de lecturas realizadas entre ambos optimizadores para esta consulta.

Para las veintidós consultas, el optimizador basado en costos realizó un total de 11,138 lecturas consistentes; que representan un 4.31% del total de lecturas consistentes hechas por ambos optimizadores.

En cambio el total de lecturas consistentes hechas por el optimizador basado en reglas para las veintidós consultas fue de 247,043; que representa el 95.69% del total de lecturas hechas por ambos optimizadores.

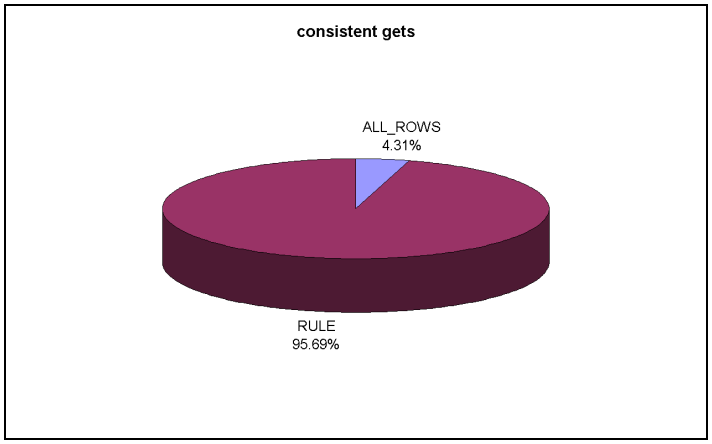

Figura 4.4 Comparación global de la estadística “consistent gets”, entre el optimizador basado en costos y basado en reglas.

4.1.2 Análisis de la estadística “physical reads”.

La comparación por consulta del número de bloques de datos leídos desde disco, la estadística “physical reads” entre el optimizador basado en costos y el optimizador basado en reglas se muestra en la siguiente gráfica.

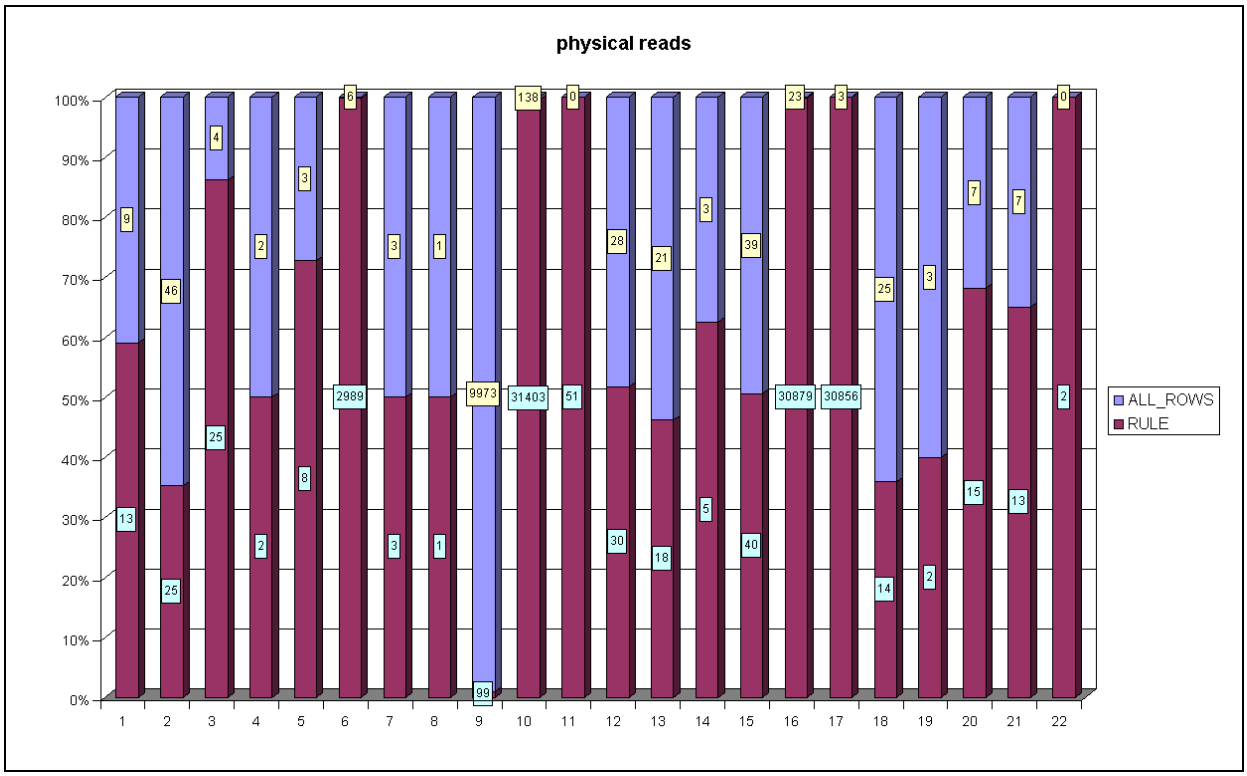

Figura 4.5 Comparación por consulta de la estadística “physical reads”, entre el optimizador basado en costos y basado en reglas.

Con el optimizador basado en reglas, la consulta con etiqueta 6 realizó 2,989 lecturas físicas, mientras que con el optimizador basado en costos llevó a cabo 6 lecturas. Las lecturas físicas realizadas con el optimizador basado en reglas para la consulta con etiqueta 6 representan más del 90% del total de lecturas realizadas entre ambos optimizadores para esta consulta. Lo mismo se observa para las consultas con etiqueta 10, 11, 16, 17 y 22.

En cambio con el optimizador basado en costos, sólo la consulta con etiqueta 9 realizó un total de lecturas físicas que representa más del 90% del total de lecturas físicas realizadas entre ambos optimizadores para esta consulta.

Para las veintidós consultas, el optimizador basado en costos realizó un total de 10,344 lecturas físicas; que representan un 9.68% del total de lecturas físicas hechas por ambos optimizadores.

En cambio el total de lecturas físicas hechas por el optimizador basado en reglas para las veintidós consultas fue de 96,493; que representa el 90.32% del total de lecturas físicas hechas por ambos optimizadores.

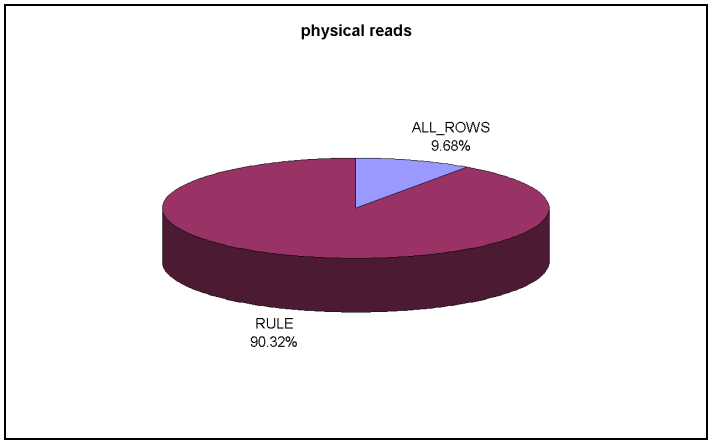

Figura 4.6 Comparación global de la estadística “physical reads”, entre el optimizador basado en costos y basado en reglas.

4.1.3 Análisis de la estadística “sorts (memory)”.

La gráfica siguiente muestra la comparación de la estadística “sorts (memory)”, ordenaciones en memoria, entre el optimizador basado en costos y basado en reglas.

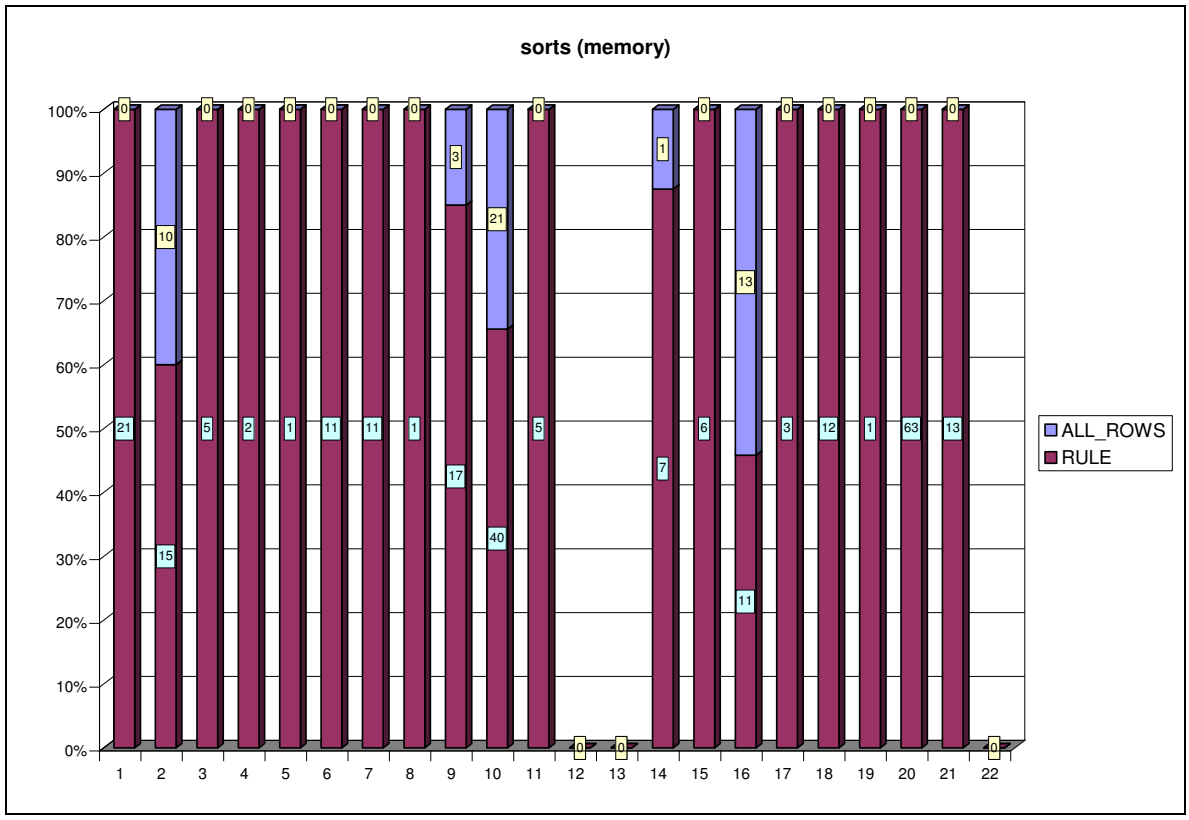

Figura 4.7 Comparación por consulta de la estadística “shorts (memory)”, entre el optimizador basado en costos y basado en reglas.

Con el optimizador basado en reglas, tres consultas no llevaron a cabo ordenaciones en memoria y las diecisiete restantes realizaron un total de 245 ordenaciones, mientras que con el optimizador basado en costos fueron diecisiete las consultas que no llevaron a cabo ordenaciones en memoria y las cinco restantes realizaron un total de 48 ordenaciones.

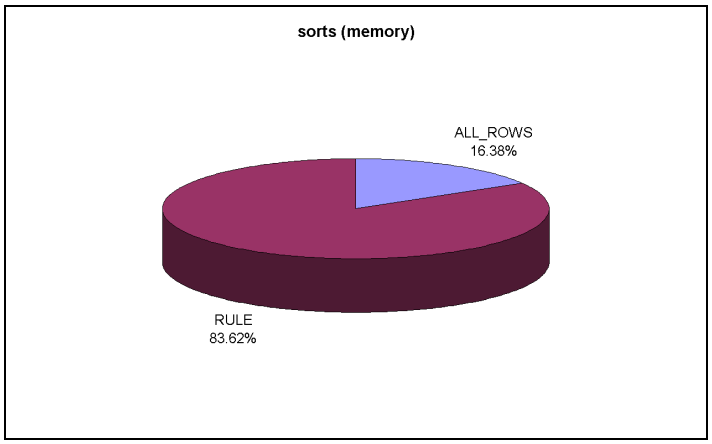

Figura 4.8 Comparación global de la estadística “sorts (memory)”, entre el optimizador basado en costos y basado en reglas.

Las ordenaciones hechas por el optimizador basado en costos representan un 16.38% del total de ordenaciones ejecutadas entre los dos optimizadores, mientras que las ordenaciones hechas por el optimizador basado en reglas representan un 83.62% del total de ordenaciones realizadas entre los dos optimizadores.

4.1.4 Selección de las consultas problemáticas y análisis de su plan de ejecución.

Con el optimizador basado en reglas, las consultas con etiqueta 6, 10, 16 y 17 tuvieron individualmente totales de lecturas consistentes y lecturas físicas de más del 90% del total de lecturas consistentes y lecturas físicas realizadas con ambos optimizadores por cada consulta.

Mientras que con el optimizador basado en costos, solamente la consulta con etiqueta 9 realizó lecturas consistentes y lecturas físicas de más del 90% del total de lecturas consistentes y lecturas físicas realizadas con ambos optimizadores para esta consulta.

A continuación se muestra el análisis de los planes de ejecución de estas consultas.

Consulta con etiqueta 6.

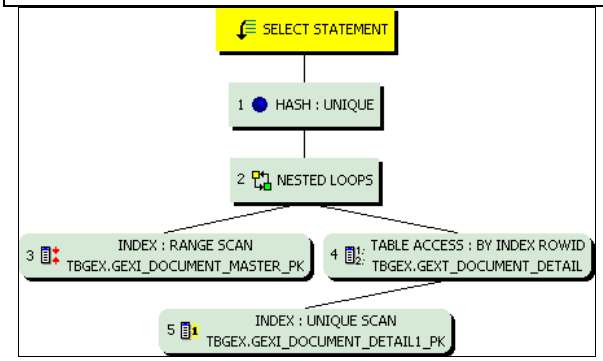

Figura 4.9 CBO: Plan de ejecución para la consulta con etiqueta 6.

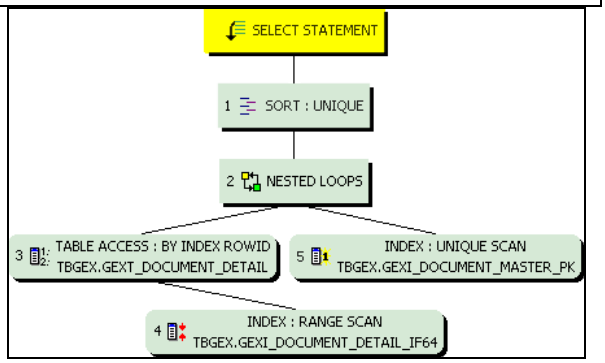

Figura 4.10 RBO: Plan de ejecución para la consulta con etiqueta 6.

El optimizador basado en costos realizó lo siguiente:

El paso 3 buscó cada DOCUMENT_ID = 'GR0125' en el índice GEXI_DOCUMENT_MASTER_PK.

El paso 5 buscó cada DOCUMENT_ID = 'GR0125', ATTRIBUTE_ID = 1516 y DOCUMENT_VERSION_ID en el índice GEXI_DOCUMENT_DETAIL1_PK y encontró los ROWIDs de los registros asociados.

El paso 4 recuperó de la tabla GEXT_DOCUMENT_DETAIL los registros con

El optimizador basado en reglas realizó lo siguiente:

El paso 4 buscó cada ATTRIBUTE_ID = 1516 en el índice GEXI_DOCUMENT_DETAIL_IF64 y encontró los ROWIDs de los registros asociados.

El paso 3 recuperó de la tabla GEXT_DOCUMENT_DETAIL los registros con los ROWIDs devueltos por el paso 4.

El paso 5 buscó cada DOCUMENT_ID y DOCUMENT_VERSION_ID en el índice GEXI_DOCUMENT_MASTER_PK filtrando

los ROWIDs devueltos por el paso 5.
El paso 2 realizó una operación en bucle anidado aceptando fuentes de registros de los pasos 3 y 4, reuniendo cada registro del paso 3 con su correspondiente registro en el paso 4, y devolviendo los registros resultantes al paso 1.
El paso 1, con los registros devueltos del paso 2 realizó una operación HASH UNIQUE para eliminar los registros duplicados, devolviéndolos al usuario emitiendo la sentencia.

INDEX_NAME	COLUMN_NAME
GEXI_DOCUMENT_DETAIL1_PK	DOCUMENT_ID
GEXI_DOCUMENT_DETAIL1_PK	DOCUMENT_VERSION_ID
GEXI_DOCUMENT_DETAIL1_PK	ATTRIBUTE_ID
GEXI_DOCUMENT_MASTER_PK	DOCUMENT_ID
GEXI_DOCUMENT_MASTER_PK	DOCUMENT_VERSION_ID

donde DOCUMENT_ID = ‘GR0125’ y encontró los ROWIDs de los registros asociados.
El paso 2 realizó una operación en bucle anidado aceptando fuentes de registros de los pasos 3 y 5, reuniendo cada registro del paso 3 con su correspondiente registro en el paso 5, y devolviendo los registros resultantes al paso 1.
El paso 1, con los registros devueltos del paso 2 realizó una operación SORT UNIQUE para eliminar los registros duplicados, devolviéndolos al usuario emitiendo la sentencia.

INDEX_NAME	COLUMN_NAME
GEXI_DOCUMENT_DETAIL_IF64	ATTRIBUTE_ID
GEXI_DOCUMENT_MASTER_PK	DOCUMENT_ID
GEXI_DOCUMENT_MASTER_PK	DOCUMENT_VERSION_ID

- El optimizador basado en costos buscó primero el documento ‘GR0125’ en la tabla maestra para después buscar su detalle para el atributo 1516.
- En cambio el optimizador basado en reglas buscó primero el detalle de todos los documentos para el atributo 1516 y después filtró el detalle del documento ‘GR0125’.

Consulta con etiqueta 10.

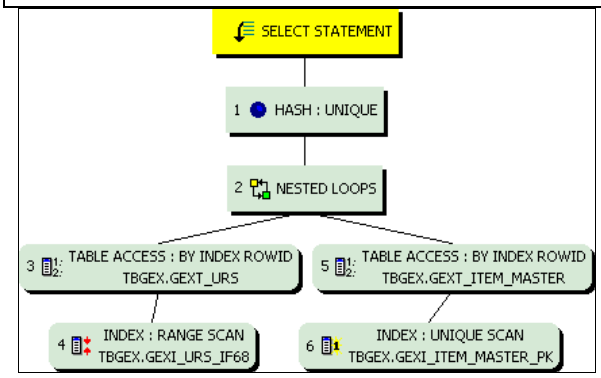

Figura 4.11 CBO: Plan de ejecución para la consulta con etiqueta 10.

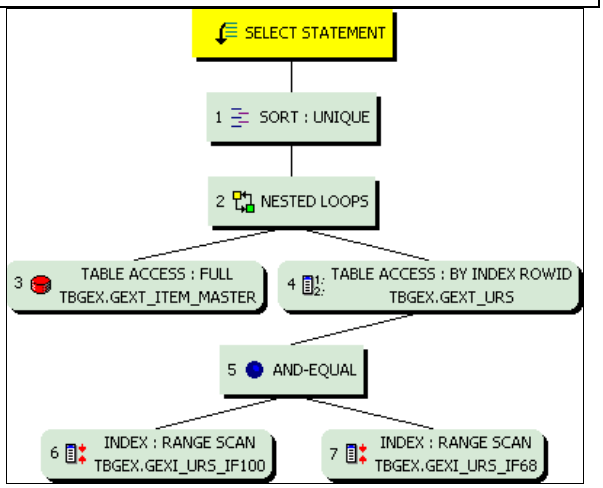

Figura 4.12 RBO: Plan de ejecución para la consulta con etiqueta 10.

El optimizador basado en costos realizó lo siguiente:
El paso 4 buscó cada UNIT_SERIAL_NUMBER = ‘282625’ en el índice GEXI_URS_IF68 y

El optimizador basado en reglas realizó lo siguiente:
El paso 6 buscó cada NEW_ITEM_NUMBER en el índice GEXI_URS_IF100 y encontró los

encontró los ROWIDs de los registros asociados.
El paso 3 recuperó de la tabla GEXT_URS los registros con los ROWIDs devueltos por el paso 4.
El paso 6 buscó cada ITEM_NUMBER en el índice GEXI_ITEM_MASTER_PK y encontró los ROWIDs de los registros asociados.
El paso 5 recuperó de la tabla GEXT_ITEM_MASTER los registros con los ROWIDs devueltos por el paso 6, filtrando donde el campo CAPITAL_PART = 1.
El paso 2 realizó una operación en bucle anidado aceptando fuentes de registros de los pasos 3 y 5, reuniendo cada registro del paso 3 con su correspondiente registro en el paso 5, y devolviendo los registros resultantes al paso 1.
El paso 1, con los registros devueltos del paso 2 realizó una operación HASH UNIQUE para eliminar los registros duplicados, devolviéndolos al usuario emitiendo la sentencia.

INDEX_NAME	COLUMN_NAME
GEXI_ITEM_MASTER_PK	ITEM_NUMBER
GEXI_URS_IF68	UNIT_SERIAL_NUMBER

ROWIDs de los registros asociados.
El paso 7 buscó cada UNIT_SERIAL_NUMBER = ‘282625’ en el índice GEXI_URS_IF168 y encontró los ROWIDs de los registros asociados.
El paso 5 realizó una operación AND-EQUAL que mezcló los ROWIDs obtenidos de los pasos 6 y 7, devolviendo un conjunto de ROWIDs que satisfacen los criterios.
El paso 4 recuperó de la tabla GEXT_URS los registros con los ROWIDs devueltos por el paso 5.
El paso 3 realizó una lectura completa de la tabla GEXT_ITEM_MASTER, filtrando donde el campo CAPITAL_PART = 1.
El paso 2 realizó una operación en bucle anidado aceptando fuentes de registros de los pasos 3 y 4, reuniendo cada registro del paso 3 con su correspondiente registro en el paso 4, y devolviendo los registros resultantes al paso 1.
El paso 1, con los registros devueltos del paso 2 realizó una operación SORT UNIQUE para eliminar los registros duplicados, devolviéndolos al usuario emitiendo la sentencia.

INDEX_NAME	COLUMN_NAME
GEXI_URS_IF100	NEW_ITEM_NUMBER
GEXI_URS_IF68	UNIT_SERIAL_NUMBER

- El optimizador basado en costos buscó primero los registros correspondientes a la turbina ‘282625’ y por cada registro devuelto realizó una operación en bucle anidado para encontrar los registros cuyo ITEM_NUMBER correspondiente sea CAPITAL_PART.
- El optimizador basado en reglas buscó primero los registros cuyo ITEM_NUMBER correspondiente sea CAPITAL_PART y por cada registro devuelto realizó una operación en bucle anidado para buscar si la parte correspondiente existe en el índice GEXI_URS_IF100 y si además está relacionada a la turbina ‘282625’.

Consulta con etiqueta 16.

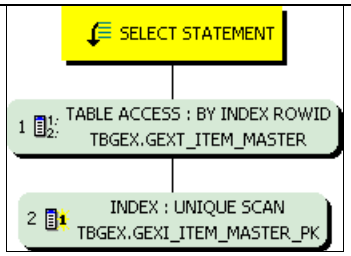

Figura 4.13 CBO: Plan de ejecución para la consulta con etiqueta 16.

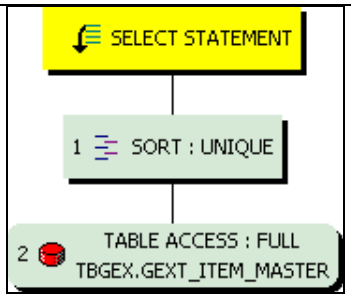

Figura 4.14 RBO: Plan de ejecución para la consulta con etiqueta 16.

El optimizador basado en costos realizó lo siguiente:

El paso 2 buscó cada ITEM_NUMBER = '109E3919G001' en el índice GEXI_ITEM_MASTER_PK y encontró el ROWID del registro asociado.

El paso 1 recuperó de la tabla GEXT_ITEM_MASTER el registro con el ROWID devuelto por el paso 2.y lo entrega al usuario emitiendo la sentencia.

INDEX_NAME	COLUMN_NAME
GEXI_ITEM_MASTER_PK	ITEM_NUMBER

El optimizador basado en reglas realizó lo siguiente:

El paso 2 realizó una lectura completa de la tabla GEXT_ITEM_MASTER, filtrando los registros donde el campo ITEM_NUMBER = '109E3919G001'.

El paso 1, con los registros devueltos del paso 2 realizó una operación SORT UNIQUE para eliminar los registros duplicados, devolviéndolos al usuario emitiendo la sentencia.

- El optimizador basado en costos realizó una lectura de índice para encontrar el registro de la parte '109E3919G001' en la tabla GEXT_ITEM_MASTER.
- El optimizador basado en reglas realizó una lectura completa de la tabla GEXT_ITEM_MASTER para después filtrar la parte '109E3919G001'.

Consulta con etiqueta 17.

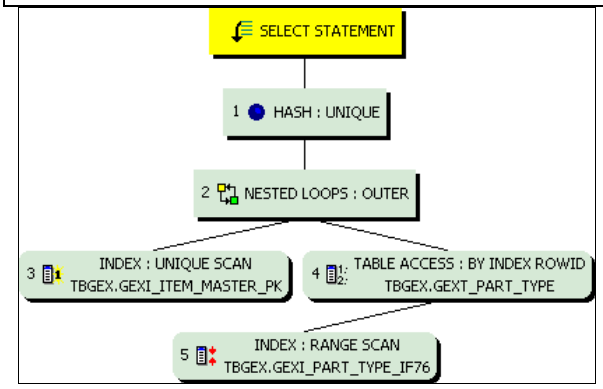

Figura 4.15 CBO: Plan de ejecución para la consulta con etiqueta 17.

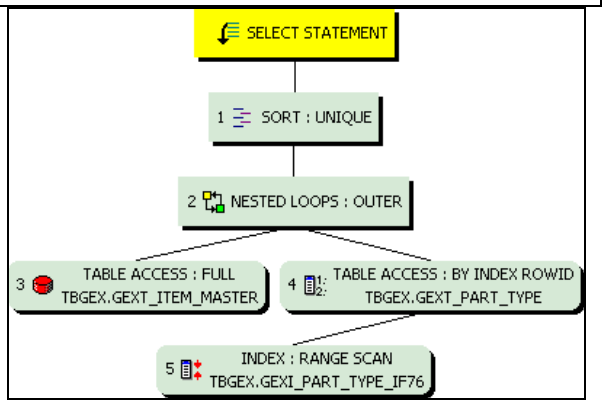

Figura 4.16 RBO: Plan de ejecución para la consulta con etiqueta 17.

El optimizador basado en costos realizó lo siguiente:

El paso 3 buscó cada ITEM_NUMBER = '109E3919G001' en el índice GEXI_ITEM_MASTER_PK y encontró el ROWID del registro asociado.

El paso 5 buscó cada ITEM_NUMBER en el índice GEXI_PART_TYPE_IF76 y encontró los ROWIDs de los registros asociados.

El paso 4 recuperó de la tabla GEXT_PART_TYPE los registros con los ROWIDs devueltos por el paso 5.

El paso 2 realizó una operación en bucle anidado externa aceptando fuentes de registros de los pasos 3 y 4, reuniendo cada registro del paso 3 con su correspondiente registro en el paso 4, aún cuando no haya registros correspondientes en el paso 4, y devolviendo los registros resultantes al paso 1.

El paso 1, con los registros devueltos del paso 2 realizó una operación HASH UNIQUE para eliminar los registros duplicados, devolviéndolos al usuario emitiendo la sentencia.

El optimizador basado en reglas realizó lo siguiente:

El paso 3 realizó una lectura completa de la tabla GEXT_ITEM_MASTER, filtrando los registros donde el campo ITEM_NUMBER = '109E3919G001'.

El paso 5 buscó cada ITEM_NUMBER en el índice GEXI_PART_TYPE_IF76 y encontró los ROWIDs de los registros asociados.

El paso 4 recuperó de la tabla GEXT_PART_TYPE los registros con los ROWIDs devueltos por el paso 5.

El paso 2 realizó una operación en bucle anidado externa aceptando fuentes de registros de los pasos 3 y 4, reuniendo cada registro del paso 3 con su correspondiente registro en el paso 4, aún cuando no haya registros correspondientes en el paso 4, y devolviendo los registros resultantes al paso 1.

El paso 1, con los registros devueltos del paso 2 realizó una operación SORT UNIQUE para eliminar los registros duplicados, devolviéndolos al usuario emitiendo la sentencia.

INDEX_NAME	COLUMN_NAME
GEXI_ITEM_MASTER_PK	ITEM_NUMBER
GEXI_PART_TYPE_IF76	ITEM_NUMBER

INDEX_NAME	COLUMN_NAME
GEXI_PART_TYPE_IF76	ITEM_NUMBER

- El optimizador basado en costos realizó una lectura de índice para encontrar el registro de la parte '109E3919G001' en la tabla GEXT_ITEM_MASTER. Después llevó a cabo una operación en bucle anidado externa para encontrar el PART_TYPE correspondiente de la parte en cuestión.
- El optimizador basado en reglas realizó una lectura completa de la tabla GEXT_ITEM_MASTER para después filtrar la parte '109E3919G001'. Después llevó a cabo una operación en bucle anidado externa para encontrar el PART_TYPE correspondiente de la parte en cuestión.

Consulta con etiqueta 9.

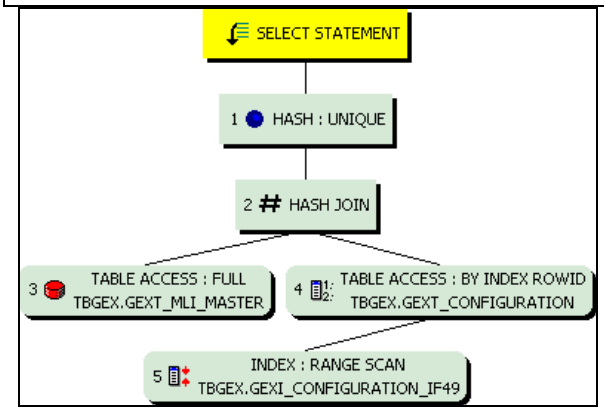

Figura 4.17 CBO: Plan de ejecución para la consulta con etiqueta 9.

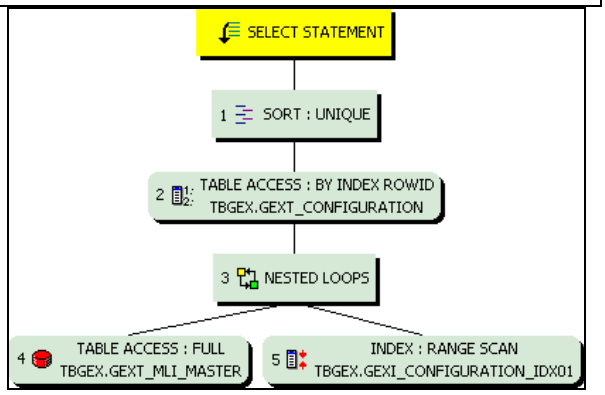

Figura 4.18 RBO: Plan de ejecución para la consulta con etiqueta 9.

El optimizador basado en costos realizó lo siguiente:

El paso 3 realizó una lectura completa de la tabla GEXT_MLI_MASTER filtrando los registros donde el campo MLI_FILTER_IND = 1.

El paso 5 buscó cada UNIT_SERIAL_NUMBER = '282625' en el índice GEXI_CONFIGURATION_IF49 y encontró los ROWIDs de los registros asociados.

El paso 4 recuperó de la tabla GEXT_CONFIGURATION los registros con los ROWIDs devueltos por el paso 5.

El paso 2 realizó una reunión por asociación,

El optimizador basado en reglas realizó lo siguiente:

El paso 4 realizó una lectura completa de la tabla GEXT_MLI_MASTER filtrando los registros donde el campo MLI_FILTER_IND = 1.

El paso 5 buscó cada UNIT_SERIAL_NUMBER = '282625 y MLI_ID en el índice GEXI_CONFIGURATION_IDX01 y encontró los ROWIDs de los registros asociados.

El paso 3 realizó una operación en bucle anidado aceptando fuentes de registros de los pasos 4 y 5, reuniendo cada registro del paso 4 con su correspondiente registro en el paso 5, y

utilizando la tabla GEXT_MLI_MASTER para construir una tabla de asociación de la llave de reunión en memoria. Después lee los registros del paso 4, probando la tabla de asociación para encontrar los registros reunidos y devolverlos al paso 1.
El paso 1, con los registros devueltos del paso 2 realizó una operación HASH UNIQUE para eliminar los registros duplicados, devolviéndolos al usuario emitiendo la sentencia.

devolviendo los registros resultantes al paso 2.
El paso 2 recuperó de la tabla GEXT_CONFIGURATION los registros con los ROWIDs devueltos por el paso 3.
El paso 1, con los registros devueltos del paso 2 realizó una operación SORT UNIQUE para eliminar los registros duplicados, devolviéndolos al usuario emitiendo la sentencia.

INDEX_NAME	COLUMN_NAME
GEXI_CONFIGURATION_IF49	UNIT_SERIAL_NUMBER

INDEX_NAME	COLUMN_NAME
GEXI_CONFIGURATION_IDX01	UNIT_SERIAL_NUMBER
GEXI_CONFIGURATION_IDX01	MLI_ID

- El optimizador basado en costos realizó una lectura completa de la tabla GEXT_MLI_MASTER filtrando los registros donde el campo MLI_FILTER_IND = 1 y con los registros filtrados construyó una tabla de asociación en memoria. Después probó esta tabla de asociación con los registros de la tabla GEXT_CONFIGURATION para la turbina '282625'.
- El optimizador basado en reglas realizó una lectura completa de la tabla GEXT_MLI_MASTER filtrando los registros donde el campo MLI_FILTER_IND = 1. Por cada registro filtrado buscó en el índice GEXI_CONFIGURATION_IDX01 para obtener los registros de la tabla GEXT_CONFIGURATION para la turbina '282625'.

4.1.4.1 Reestructuración de una consulta problemática para su afinación.

Ya que la consulta con etiqueta 9 tuvo un mejor desempeño bajo el enfoque de optimización basado en reglas, se modificó el texto de esta consulta, añadiéndole un hint para sugerir entonces al optimizador basado en costos realizar una reunión en bucle anidado en lugar de una reunión por asociación entre las dos fuentes de registros.

El texto de la consulta quedó de la siguiente manera:

```
SELECT /*+ USE_NL(MLI CFG) */ DISTINCT CFG.*
  FROM GEXV_CONFIGURATION CFG, GEXT_MLI_MASTER MLI
 WHERE (    (MLI.MLI_ID = CFG.MLITYPE)
        AND (CFG.EQUIPMENT = '282625')
        AND (MLI.MLI_FILTER_IND = 1));
```

Se ejecutó la consulta modificada mediante SQL*Plus AUTOTRACE con el enfoque de optimización basado en costos y sus nuevas estadísticas se muestran en la tabla siguiente:

Query	Elapsed	Plan hash value	recursive calls	db block gets	consistent gets	physical reads	redo size	bytes sent via SQL*Net to client	bytes received via SQL*Net from client	SQL*Net roundtrips to/from client	sorts (memory)	sorts (disk)	rows processed
1	00:00:01.43	1541948779	1	0	61	9	0	422	246	3	0	0	20
2	00:00:01.54	2323000060	784	0	1227	25	0	565	239	2	15	0	4
3	00:00:01.21	2612966111	15	0	9	4	0	354	239	2	0	0	6
4	00:00:01.18	2122101414	1	0	2	2	0	445	239	2	0	0	1
5	00:00:01.31	3508301604	1	0	9	3	0	464	246	3	0	0	24
6	00:00:01.23	2035648940	15	0	11	6	0	247	239	2	0	0	1
7	00:00:01.32	1049795864	1	0	9	3	0	260	239	2	0	0	2
8	00:00:01.17	719383829	1	0	4	1	0	428	239	2	0	0	1
9	**00:00:02.23**	**3217695679**	**8**	**0**	**175**	**99**	**0**	**1556**	**252**	**4**	**0**	**0**	**34**
10	00:00:02.28	3324069663	706	0	456	138	0	1267	239	2	21	0	14
11	00:00:01.46	642984172	88	0	102	0	0	2626	253	4	0	0	39
12	00:00:01.54	4258698081	1	0	32	28	0	1428	246	3	0	0	27
13	00:00:01.51	1548254942	8	0	25	21	0	1395	246	3	0	0	22
14	00:00:01.18	1095053854	1	0	3	3	0	282	239	2	1	0	2
15	00:00:01.76	2148511359	1	0	47	39	0	930	260	5	0	0	51
16	00:00:01.46	3185817551	117	0	64	23	0	547	239	2	13	0	1
17	00:00:01.18	316242083	1	0	6	3	0	246	239	2	0	0	1
18	00:00:01.34	1043000449	60	0	39	25	0	255	239	2	0	0	3
19	00:00:01.18	3770764186	15	0	12	3	0	239	239	2	0	0	1
20	00:00:01.26	3328739327	92	0	49	7	0	407	239	2	0	0	1
21	00:00:01.31	1040190692	15	0	13	7	0	248	239	2	0	0	2
22	00:00:01.15	4097211151	1	0	4	0	0	241	239	2	0	0	1

Tabla 4.3 Estadísticas de SQL*Plus AUTOTRACE - Optimizador Basado en Costos con la consulta con etiqueta 9 modificada.

4.1.4.2 Comparación de las estadísticas “consistent gets”, “physical reads” y “sorts (memory)” entre la consulta con etiqueta 9 original y la consulta con etiqueta 9 con hint, ambas ejecutadas con el CBO.

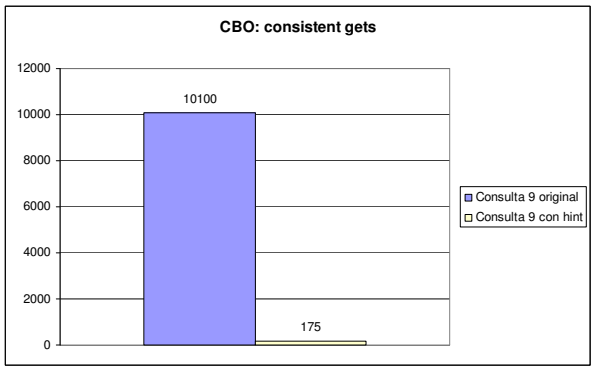

Figura 4.19 CBO: consistent gets, entre la consulta con etiqueta 9 original y con hint.

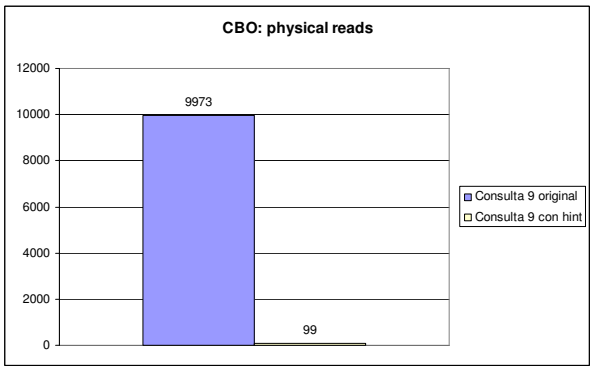

Figura 4.20 CBO: physical reads, entre la consulta con etiqueta 9 original y con hint.

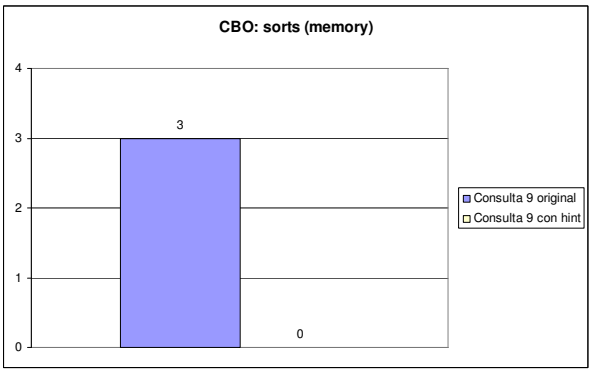

Figura 4.21 CBO: shorts (memory), entre la consulta con etiqueta 9 original y con hint.

Originalmente para la consulta con etiqueta 9 con el optimizador basado en costos se tuvieron 10,100 lecturas consistentes, 9,973 lecturas físicas y 3 ordenaciones en memoria.

Para la consulta con etiqueta 9 con hint se tuvieron ahora 175 lecturas consistentes, 99 lecturas físicas y ninguna ordenación en memoria.

Se observa que hubo una mejora significativa en las tres estadísticas, del 98.27% para lecturas consistentes, 99.01% para lecturas físicas y del 100% para ordenaciones en memoria.

La consulta con etiqueta 9 modificada invocará siempre al optimizador basado en costos ya que contiene un hint como parte de la sentencia.

El nuevo plan de ejecución obtenido y su análisis se muestran en la siguiente figura.

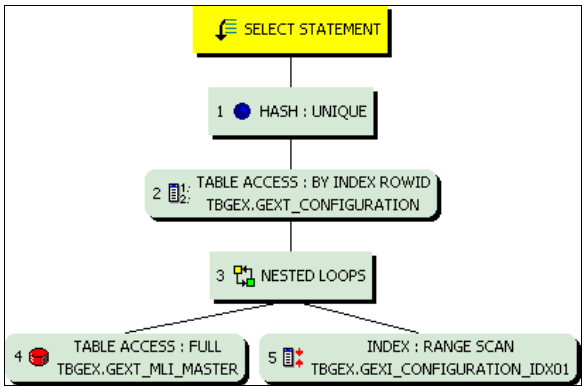

Figura 4.22 CBO: Plan de ejecución para la consulta con etiqueta 9 modificada.

El optimizador basado en costos realizó lo siguiente:

El paso 4 realizó una lectura completa de la tabla GEXT_MLI_MASTER filtrando los registros donde el campo MLI_FILTER_IND = 1.
El paso 5 buscó cada UNIT_SERIAL_NUMBER = '282625' y MLI_ID en el índice GEXI_CONFIGURATION_IDX01 y encontró los ROWIDs de los registros asociados.
El paso 3 realizó una operación en bucle anidado aceptando fuentes de registros de los pasos 4 y 5, reuniendo cada registro del paso 4 con su correspondiente registro en el paso 5, y devolviendo los registros resultantes al paso 2.
El paso 2 recuperó de la tabla GEXT_CONFIGURATION los registros con los ROWIDs devueltos por el paso 3.
El paso 1, con los registros devueltos del paso 2 realizó una operación HASH UNIQUE para eliminar los registros duplicados, devolviéndolos al usuario emitiendo la sentencia.

INDEX_NAME	COLUMN_NAME
GEXI_CONFIGURATION_IDX01	UNIT_SERIAL_NUMBER
GEXI_CONFIGURATION_IDX01	MLI_ID

4.1.4.3 Comparación de las estadísticas "consistent gets", "physical reads" y "sorts (memory)" entre la consulta con etiqueta 9 original ejecutada con el RBO y la consulta con etiqueta 9 con hint, ejecutada con el CBO.

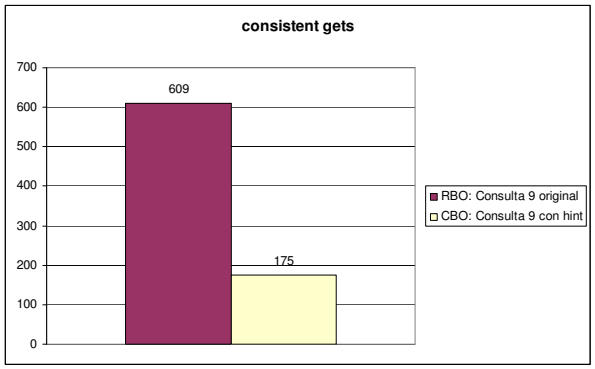

Figura 4.23 Comparación de la estadística "consistent gets", entre el RBO con la consulta con etiqueta 9 original y el CBO con la consulta con hint.

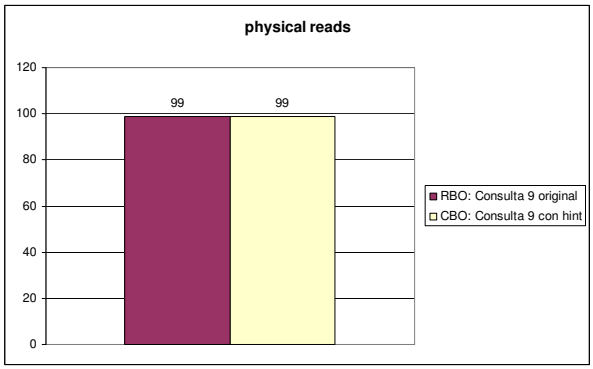

Figura 4.24 Comparación de la estadística "physical reads", entre el RBO con la consulta con etiqueta 9 original y el CBO con la consulta con hint.

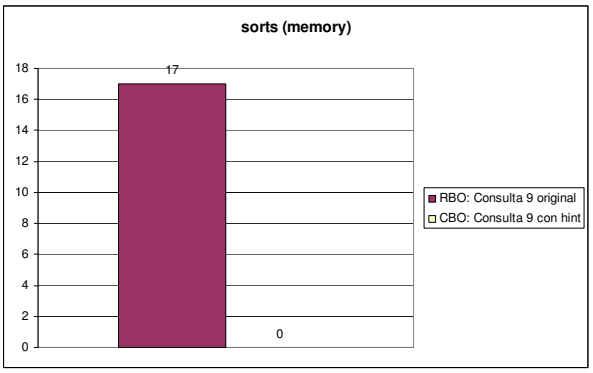

Figura 4.25 Comparación de la estadística “shorts (memory)”, entre el RBO con la consulta con etiqueta 9 original y el CBO con la consulta con hint.

Con el optimizador basado en reglas la consulta con etiqueta 9 original realizó 609 lecturas consistentes mientras que con el optimizador basado en costos la consulta con hint realizó 175 lecturas consistentes, esto representa una mejora del 71.26% con el optimizador basado en costos.

Para las lecturas físicas ambos optimizadores hicieron el mismo número de lecturas, que fueron 99; y en cuanto a ordenaciones en memoria la mejora fue del 100%, ya que de 17 ordenaciones que realizó el optimizador basado en reglas, el optimizador basado en costos no realizó ninguna con la consulta con hint.

4.2 Afinación automática con SQL Tuning Advisor.

De igual manera, se sometieron las consultas al optimizador de afinación automático disponible en Oracle 10g para el enfoque de optimización basado en costos, mediante la herramienta SQL Tuning Advisor.

4.2.1 Creación de tareas de afinación.

Para ello se creó una tabla de nombre GEXT_QUERY_LIST, con dos campos, QUERY_ID de tipo NUMBER y QUERY_TEXT de tipo CLOB. El campo QUERY_ID es un identificador consecutivo que se corresponde con el número de etiqueta utilizado por la consulta dentro del script sql_queries.sql. El campo QUERY_TEXT contiene el texto de la consulta.

Se insertaron las veintidós consultas en la tabla y por cada una se creó una tarea de afinación mediante el script create_sql_tuning_tasks.sql que se muestra a continuación:

```
DECLARE
   MY_TASK_NAME   VARCHAR2 (30);

   TYPE GENERIC_CURTYPE IS REF CURSOR;

   C_QUERIES      GENERIC_CURTYPE;
BEGIN
   FOR C_QUERIES IN (SELECT   *
                       FROM GEXT_QUERY_LIST
                   ORDER BY QUERY_ID) LOOP
      MY_TASK_NAME :=
         DBMS_SQLTUNE.CREATE_TUNING_TASK
         -- El texto de la sentencia SQL
         (SQL_TEXT        => C_QUERIES.QUERY_TEXT,
         -- El nombre de usuario para quien será afinada la sentencia
         USER_NAME       => 'TBGEX',
         -- Alcance de la afinación (limited/comprehensive)
         SCOPE           => 'LIMITED',
         -- La duración máxima en segundos para la sesión de afinación
         TIME_LIMIT      => 60,
         -- Un nombre opcional para la tarea de afinación
         TASK_NAME       =>    'TUNING_TASK_'
         || C_QUERIES.QUERY_ID);
   END LOOP;
END;
```

El script create_sql_tuning_tasks.sql recorre la tabla GEXT_QUERY_LIST y por cada consulta, crea una tarea de afinación invocando al procedimiento DBMS_SQLTUNE.CREATE_TUNING_TASK. Este procedimiento recibe como parámetros: la consulta, el nombre de usuario para el cual será afinada la sentencia, el alcance de la tarea de afinación, la duración máxima en segundos para la sesión de afinación y el nombre de la tarea de afinación.

El alcance de la tarea de afinación se definió como LIMITED para todos los casos de forma que SQL Tuning Advisor realice únicamente verificaciones de estadísticas, análisis de rutas de acceso, y análisis de estructura SQL.

El nombre de la tarea de afinación se definió como la cadena 'TUNING_TASK_' concatenada con el valor del campo QUERY_ID. El nombre de la tarea de afinación puede utilizarse para consultar el estatus de las tareas en la vista del diccionario de datos USER_ADVISOR_TASKS.

TASK_ID	TASK_NAME	ADVISOR_NAME	CREATED	LAST_MODIFIED	EXECUTION_START	EXECUTION_END	STATUS
8786	TUNING_TASK_1	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8787	TUNING_TASK_2	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8788	TUNING_TASK_3	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8789	TUNING_TASK_4	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8790	TUNING_TASK_5	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8791	TUNING_TASK_6	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8792	TUNING_TASK_7	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8793	TUNING_TASK_8	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8794	TUNING_TASK_9	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8795	TUNING_TASK_10	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8796	TUNING_TASK_11	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8797	TUNING_TASK_12	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8798	TUNING_TASK_13	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8799	TUNING_TASK_14	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8800	TUNING_TASK_15	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8801	TUNING_TASK_16	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8802	TUNING_TASK_17	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8803	TUNING_TASK_18	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8803	TUNING_TASK_18	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8804	TUNING_TASK_19	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8805	TUNING_TASK_20	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8806	TUNING_TASK_21	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED
8807	TUNING_TASK_22	SQL Tuning Advisor	4/2/2009 9:47	4/2/2009 9:48	4/2/2009 9:48	4/2/2009 9:48	COMPLETED

Figura 4.26 Tareas de afinación en la vista USER_ADVISOR_TASKS.

4.2.2 Ejecución de tareas de afinación.

Se realizó la ejecución de cada tarea de afinación SQL mediante el script exec_sql_tuning_tasks.sql

```
DECLARE
   loop_counter   PLS_INTEGER;
BEGIN
   FOR loop_counter IN 1 .. 22 LOOP
      DBMS_SQLTUNE.EXECUTE_TUNING_TASK (task_name      =>    'TUNING_TASK_'
                                                          || loop_counter);
   END LOOP;
END;
```

4.2.3 Resultados de tareas de afinación.

Los resultados de afinación se obtuvieron con la consulta siguiente sobre ciertas vistas del diccionario de datos:

```
SELECT   ADVTK.TASK_ID, ADVTK.TASK_NAME, ADVFND.FINDING_ID, ADVFND.TYPE,
         ADVFND.MESSAGE FINDING_MSG, ADVACT.ACTION_ID, ADVACT.COMMAND,
         ADVACT.ATTR1, ADVACT.ATTR3, ADVACT.ATTR5,
         ADVACT.MESSAGE RECOMMENDATION_MSG, ADVRAT.MESSAGE RATIONALE_MSG
    FROM SYS.USER_ADVISOR_TASKS ADVTK,
         SYS.USER_ADVISOR_FINDINGS ADVFND,
         SYS.USER_ADVISOR_RECOMMENDATIONS ADVREC,
         SYS.USER_ADVISOR_ACTIONS ADVACT,
         SYS.USER_ADVISOR_RATIONALE ADVRAT
   WHERE (    (ADVTK.TASK_ID = ADVFND.TASK_ID(+))
          AND (ADVTK.TASK_ID = ADVREC.TASK_ID(+))
          AND (ADVTK.TASK_ID = ADVACT.TASK_ID(+))
          AND (ADVTK.TASK_ID = ADVRAT.TASK_ID(+)))
ORDER BY ADVTK.TASK_ID, ADVACT.ACTION_ID ASC
```

Los resultados de cada tarea de afinación se muestran a continuación:

TASK_ID	TASK_NAME	FINDING_ID	TYPE	FINDING_MSG	ACTION_ID	COMMAND	ATTR1	ATTR3	ATTR5	RECOMMENDATION_MSG	RATIONALE_MSG
8786	TUNING_TASK_1										
8787	TUNING_TASK_2	1	PROBLEM	The execution plan of this statement can be improved by creating one or more indices.	1	CREATE INDEX	TBGEX.IDX$$_22530001	TBGEX.GEXT_CONFIGURATION	("PARENT_BMFL_SEQUENCE","UNIT_SERIAL_NUMBER")	Consider running the Access Advisor to improve the physical schema design or creating the recommended index.	Creating the recommended indices significantly improves the execution plan of this statement. However, it might be preferable to run "Access Advisor" using a representative SQL workload as opposed to a single statement. This will allow to get comprehensive index recommendations which takes into account index maintenance overhead and additional space consumption.
8788	TUNING_TASK_3	1	PROBLEM	The execution plan of this statement can be improved by creating one or more indices.	1	CREATE INDEX	TBGEX.IDX$$_22540001	TBGEX.GEXT_CAPABILITY_DETAIL	("CAPABILITY_ID","DOCUMENT_VERSION_ID","FOLDER_ID")	Consider running the Access Advisor to improve the physical schema design or creating the recommended index.	Creating the recommended indices significantly improves the execution plan of this statement. However, it might be preferable to run "Access Advisor" using a representative SQL workload as opposed to a single statement. This will allow to get comprehensive index recommendations which takes into account index maintenance overhead and additional space consumption.
8789	TUNING_TASK_4										
8790	TUNING_TASK_5										
8791	TUNING_TASK_6	1	PROBLEM	The execution plan of this statement can be improved by creating one or more indices.	1	CREATE INDEX	TBGEX.IDX$$_22570001	TBGEX.GEXT_DOCUMENT_DETAIL	("DOCUMENT_ID","ATTRIBUTE_ID")	Consider running the Access Advisor to improve the physical schema design or creating the recommended index.	Creating the recommended indices significantly improves the execution plan of this statement. However, it might be preferable to run "Access Advisor" using a representative SQL workload as opposed to a single statement. This will allow to get comprehensive index recommendations which takes into account index maintenance overhead and additional space consumption.
8792	TUNING_TASK_7										
8793	TUNING_TASK_8	1	PROBLEM	The execution plan of this statement can be improved by creating one or more indices.	1	CREATE INDEX	TBGEX.IDX$$_22590001	TBGEX.GEXT_DOCUMENT_MASTER	("DOCUMENT_ID")	Consider running the Access Advisor to improve the physical schema design or creating the recommended index.	Creating the recommended indices significantly improves the execution plan of this statement. However, it might be preferable to run "Access Advisor" using a representative SQL workload as opposed to a single statement. This will allow to get comprehensive index recommendations which takes into account index maintenance overhead and additional space consumption.
8794	TUNING_TASK_9										
8795	TUNING_TASK_10										
8796	TUNING_TASK_11										
8797	TUNING_TASK_12										
8798	TUNING_TASK_13										
8799	TUNING_TASK_14										
8800	TUNING_TASK_15										
8801	TUNING_TASK_16										
8802	TUNING_TASK_17										
8803	TUNING_TASK_18	1	PROBLEM	The execution plan of this statement can be improved by creating one or more indices.	1	CREATE INDEX	TBGEX.IDX$$_22630001	TBGEX.GEXT_CAPABILITY_ITEM_TYPE	("ITEM_NUMBER")	Consider running the Access Advisor to improve the physical schema design or creating the recommended index.	Creating the recommended indices significantly improves the execution plan of this statement. However, it might be preferable to run "Access Advisor" using a representative SQL workload as opposed to a single statement. This will allow to get comprehensive index recommendations which takes into account index maintenance overhead and additional space consumption.
8803	TUNING_TASK_18	1	PROBLEM	The execution plan of this statement can be improved by creating one or more indices.	2	CREATE INDEX	TBGEX.IDX$$_22630002	TBGEX.GEXT_ITEM_CRITERIA	("CAPABILITY_ITEM_ID")	Consider running the Access Advisor to improve the physical schema design or creating the recommended index.	Creating the recommended indices significantly improves the execution plan of this statement. However, it might be preferable to run "Access Advisor" using a representative SQL workload as opposed to a single statement. This will allow to get comprehensive index recommendations which takes into account index maintenance overhead and additional space consumption.
8804	TUNING_TASK_19										
8805	TUNING_TASK_20										
8806	TUNING_TASK_21	1	PROBLEM	The execution plan of this statement can be improved by creating one or more indices.	1	CREATE INDEX	TBGEX.IDX$$_22660001	TBGEX.GEXT_PERFORMANCE_DOCUMENT	("UNIT_SERIAL_NUMBER","DOCUMENT_ID")	Consider running the Access Advisor to improve the physical schema design or creating the recommended index.	Creating the recommended indices significantly improves the execution plan of this statement. However, it might be preferable to run "Access Advisor" using a representative SQL workload as opposed to a single statement. This will allow to get comprehensive index recommendations which takes into account index maintenance overhead and additional space consumption.
8807	TUNING_TASK_22										

Figura 4.27 Resultados de afinación para las veintidós consultas con la herramienta SQL Tuning Advisor de Oracle.

Se observa que la herramienta sugiere principalmente la creación de índices para las consultas con etiquetas 2, 3, 6, 8, 18 y 21; así como la ejecución de la herramienta “Access Advisor” con una carga de trabajo SQL representativa en vez de una sola sentencia.

Las consultas que obtuvieron recomendaciones realizaron un total de 157 lecturas consistentes, que representan el 1.41% del total de lecturas hechas por todas las consultas con el enfoque de optimización basado en costos.

En cuanto a lecturas físicas y ordenaciones en memoria, estas consultas realizaron un total de 89 lecturas y 10 ordenaciones, que representan respectivamente el 0.86% y el 20.83% del total de lecturas físicas y ordenaciones realizadas por todas las consultas con el enfoque de optimización basado en costos.

Cabe notar que no se obtuvo ninguna recomendación para la consulta con etiqueta 9 cuyas lecturas en modo consistente representaron el 90.68% del total de lecturas llevadas a cabo por todas las consultas con el optimizador basado en costos, mientras que sus lecturas físicas representaron el 96.41% del total de lecturas hechas por todas las consultas, y cuyas ordenaciones representaron el 6.25% del total de ordenaciones realizadas por todas las consultas.

Ya que el total de lecturas consistentes, lecturas físicas y ordenaciones de las consultas con recomendaciones no representan un porcentaje significativo del total correspondiente obtenido por todas las consultas con el enfoque de optimización basado en costos, y considerando además que la creación de índices requiere la evaluación de cuestiones como espacio en disco y en memoria, o la sobrecarga que involucra mantener un índice, se decidió no seguir las recomendaciones de crear los índices sugeridos.

CONCLUSIONES Y RECOMENDACIONES PARA TRABAJOS FUTUROS.

El caso práctico permitió conocer las estructuras y configuraciones de un servidor de bases de datos relacional Oracle y la forma cómo intervienen durante el procesamiento de sentencias SQL.

Se observó también el comportamiento del optimizador de Oracle en sus dos enfoques, basado en costos y basado en reglas, durante el procesamiento de un conjunto de consultas idéntico.

Se manejaron las estadísticas de base de datos empleadas por el optimizador basado en costos para generar las descripciones de tablas, índices y columnas que estarían disponibles durante la ejecución del conjunto de consultas definido.

Se describió que el optimizador basado en costos cuenta con más rutas de acceso y herramientas para apoyar la tarea de afinación de sentencias SQL, a diferencia del optimizador basado en reglas que posee solamente un conjunto predefinido de rutas de acceso clasificadas de forma heurística para el procesamiento de consultas.

El análisis de los resultados obtenidos muestra que el optimizador basado en costos tuvo un mejor desempeño al generar planes de ejecución cuyos totales por conjunto de consultas para la estadística de lecturas consistentes representa el 4.31% del total de lecturas hechas por ambos optimizadores, para las lecturas físicas representa el 9.68% del total de lecturas realizadas por ambos optimizadores, mientras que para las ordenaciones en memoria el total por conjunto de consultas fue del 16.38% del total de ordenaciones llevadas a cabo por ambos optimizadores.

El optimizador basado en reglas tuvo un desempeño más pobre ya que:

- En las lecturas consistentes, quince de las veintidós consultas presentaron un total de lecturas mayor al 80% del total de lecturas hechas por cada consulta bajo ambos enfoques de optimización. Mientras que con el optimizador basado en costos, solo una consulta tuvo un total de lecturas que representa más del 80% del total de lecturas obtenidas con ambos optimizadores.

- Para las lecturas físicas, siete de las veintidós consultas presentaron un total de lecturas mayor al 80% del total de lecturas hechas por cada consulta bajo ambos enfoques de optimización. Mientras que con el optimizador basado en costos, solo una consulta tuvo un total de lecturas físicas que representa más del 80% del total de lecturas obtenidas con ambos enfoques de optimización.
- En cuanto a las ordenaciones en memoria, dieciséis de las veintidós consultas presentaron un total de ordenaciones mayor al 80% del total de ordenaciones hechas por cada consulta bajo ambos enfoques de optimización. Mientras que con el optimizador basado en costos solo cinco consultas efectuaron ordenaciones en memoria.

Se mostró un caso de afinación para la consulta con etiqueta 9, que fue la que tuvo el peor desempeño con el optimizador basado en costos. Modificando la consulta al agregarle un hint, se mejoraron significativamente los resultados originales obtenidos con el optimizador basado en costos. Estos nuevos resultados para la consulta con etiqueta 9 modificada, también fueron mejores que los resultados originales obtenidos con el optimizador basado en reglas.

Se sometieron de igual manera las consultas al optimizador de afinación automático que acompaña a Oracle 10g mediante la creación y ejecución de tareas de afinación empleando el paquete DBMS_SQLTUNE. Los resultados obtenidos recomendaban la creación de nuevos índices para mejorar las rutas de acceso a los datos. Sin embargo se decidió no crear ningún índice ya que las recomendaciones aplicaban a consultas cuyas estadísticas de lecturas consistentes y lecturas físicas representaban únicamente el 1.41% y el 0.86% del total de lecturas hechas por el conjunto de consultas definido bajo el optimizador basado en costos.

Por lo anterior, se concluye que el enfoque de optimización basado en costos mejora el uso de recursos computacionales y en consecuencia el tiempo de respuesta durante la ejecución de consultas en bases de datos relacionales empleadas por aplicaciones OLTP.

Como posibles temas de investigación para trabajos futuros se sugiere:

- Otras herramientas de optimización disponibles en Oracle 10g como SQL Trace y TKPROF.
- Cómo utilizar hints, histogramas y planes de ejecución almacenados para la optimización basada en costos.
- Cómo se comportan el optimizador basado en costos y el optimizador basado en reglas en aplicaciones DSS.
- Cuáles son los enfoques de optimización disponibles en otros manejadores de base de datos.
- Extrapolar el tema de optimización de consultas a bases de datos distribuidas.

LITERATURA CITADA

[1] Oracle Support. (2002, Junio). 10626.1 Cost Based Optimizer (CBO) Overview. Recuperado de http://metalink.oracle.com/metalink/plsql/showdoc?db=NOT&id=10626.1

[2] Microsoft Corporation. (2002, Febrero). Query Optimization Techniques: Contrasting Various Optimizer Implementations with Microsoft SQL Server. Recuperado de http://msdn.microsoft.com/archive/default.asp?url=/archive/enus/dnarsqlsg/html/msdn_qryoptim.asp

[3] Jacobs, K. (2002, Julio/Agosto). Query Optimization. Oracle Magazine. Recuperado de http://www.oracle.com/technology/oramag/oracle/02-jul/index.html

[4] Silberschatz, A., Korth, F. (2002). Procesamiento de Consultas. En C. Fernández (Ed.), *Fundamentos de bases de datos* (pp. 319-341). España: Mc Graw Hill.

[5] Floss, K. (2005, Enero/Febrero). Understanding Optimization. Oracle Magazine. Recuperado de http://www.oracle.com/technology/oramag/oracle/05-jan/o15tech_tuning.html

[6] Burleson, D. (2003). Cost Control: Inside the Oracle Optimizer. Oracle Corporation. Recuperado de http://www.oracle.com/technology/oramag/webcolumns/2003/techarticles/burleson_cbo_pt1.html

[7] Ziauddin, M., Zait, M., Minhas, M., Yagoub, K. (2006, Junio). Introduction. *The Self-Managing Database: Guided Application & SQL Tuning with Oracle Database 10g Release 2 [An Oracle White Paper]*. (pp. 3-4). Estados Unidos: Oracle Corporation.

[8] Connolly, T., Begg, E. (2005). Query Processing. En Pearson Education Limited (Ed.), *Database systems: a practical approach to design, implementation, and management* (pp. 630-682). Inglaterra: Addison-Wesley.

[9] Silberschatz, A., Korth, F. (2002). Optimización de Consultas. En C. Fernández (Ed.), *Fundamentos de bases de datos* (pp. 343-364). España: Mc Graw Hill.

[10] Cyran, M. (1999, Diciembre). Understanding Oracle Performance Tuning. *Oracle8i Designing and Tuning for Performance Release 2 (8.1.6)* (pp. 2-6). Estados Unidos: Oracle Corporation.

[11] Vennapusa, P., (2003). Following a Tuning Methodology. Understanding Scalability. En N. Brozowski (Ed.), *Oracle 9i Database: SQL Tuning Workshop R2*. (pp. 10-11). Estados Unidos: Oracle Corporation.

[12] Vennapusa, P., (2003). Following a Tuning Methodology. Tuning Roles. En N. Brozowski (Ed.), *Oracle 9i Database: SQL Tuning Workshop R2*. (p. 15). Estados Unidos: Oracle Corporation.

[13] Vennapusa, P., (2003). SQL Statement Processing. System Global Area. En N. Brozowski (Ed.), *Oracle 9i Database: SQL Tuning Workshop R2*. (p. 4). Estados Unidos: Oracle Corporation.

[14] Chan, I. (2008, Marzo). Memory Configuration and Use. Understanding Memory Allocation Issues. Automatic Shared Memory Management. *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*. (p. 2). Estados Unidos: Oracle.

[15] Vennapusa, P., (2003). SQL Statement Processing. Shared SQL Areas. En N. Brozowski (Ed.), *Oracle 9i Database: SQL Tuning Workshop R2*. (pp. 6-7). Estados Unidos: Oracle Corporation.

[16] Green, C. (2002, Octubre). Introduction to the Optimizer. Overview of SQL Processing. *Oracle9i Database Performance Tuning Guide and Reference Release 2 (9.2)*. (p. 2). Estados Unidos: Oracle Corporation.
[17] Cyran, M., Lane, P., Polk, J.P. (2005, Octubre). Memory Architecture. Overview of the System Global Area. Shared Pool. Shared SQL Areas and Private SQL Areas. *Oracle Database Concepts 10g Release 2 (10.2)*. (p. 10). Estados Unidos: Oracle.
[18] Vennapusa, P., (2003). SQL Statement Processing. SQL Statement Processing Phases. En N. Brozowski (Ed.), *Oracle 9i Database: SQL Tuning Workshop R2*. (pp. 10-15). Estados Unidos: Oracle Corporation.
[19] Green, C. (2002, Octubre). Using Autotrace in SQL*Plus. Overview of the Autotrace Report. *Oracle9i Database Performance Tuning Guide and Reference Release 2 (9.2)*. (pp. 2-4). Estados Unidos: Oracle Corporation.
[20] Chan, I. (2008, Marzo). The Query Optimizer. Understanding the Query Optimizer. *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*. (pp. 11-13). Estados Unidos: Oracle.
[21] Kyte, T. (2003, Enero/Febrero). On the Explicit, Size, and Complex. Oracle Magazine. Recuperado de http://www.oracle.com/technology/oramag/oracle/03-jan/o13asktom.html
[22] Cyran, M. (1999, Diciembre). The Optimizer. What is the Optimizer? *Oracle8i Designing and Tuning for Performance Release 2 (8.1.6)* (pp. 4-5). Estados Unidos: Oracle Corporation.
[23] Floss, K. (2005, Enero/Febrero). Understanding Optimization. Oracle Magazine. Recuperado de http://www.oracle.com/technology/oramag/oracle/05-jan/o15tech_tuning.html
[24] Chan, I. (2008, Marzo). The Query Optimizer. Understanding Access Paths for the Query Optimizer. How the Query Optimizer Chooses an Access Path. *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*. (p. 22). Estados Unidos: Oracle.
[25] Green, C. (2002, Octubre). Introduction to the Optimizer. Understanding the Cost-Based Optimizer. *Oracle9i Database Performance Tuning Guide and Reference Release 2 (9.2)*. (p. 10). Estados Unidos: Oracle Corporation.
[26] Green, C. (2002, Octubre). Introduction to the Optimizer. Choosing an Optimizer Approach and Goal. *Oracle9i Database Performance Tuning Guide and Reference Release 2 (9.2)*. (pp. 5-8). Estados Unidos: Oracle Corporation.
[27] Chan, I. (2008, Marzo). The Query Optimizer. Understanding Access Paths for the Query Optimizer. *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*. (pp. 13-21). Estados Unidos: Oracle.
[28] Chan, I. (2008, Marzo). The Query Optimizer. Understanding Joins. *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*. (pp. 22-30).
[29] Chan, I. (2008, Marzo). The Query Optimizer. Enabling and Controlling Query Optimizer Features. *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*. (pp. 4-6). Estados Unidos: Oracle.
[30] Chan, I. (2008, Marzo). Memory Configuration and Use. PGA Memory Management. *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*. (pp. 37-38). Estados Unidos: Oracle.
[31] Knowledge Xpert for PL/SQL 10.0 SQL Optimizers – Rule-Based
[32] Green, C. (2002, Octubre). Using the Rule-Based Optimizer. Overview of the Rule-Based Optimizer (RBO). *Oracle9i Database Performance Tuning Guide and Reference Release 2 (9.2)*. (pp. 1-2). Estados Unidos: Oracle Corporation.

[33] Green, C. (2002, Octubre). Using the Rule-Based Optimizer. Understanding Access Paths for the RBO. *Oracle9i Database Performance Tuning Guide and Reference Release 2 (9.2)*. (pp. 3-15). Estados Unidos: Oracle Corporation.
[34] Burleson, D. The Cost-Based Optimizer (CBO). Recuperado de http://www.remote-dba.net/t_op_sql_cost_based_optimizers.htm
[35] Burleson, D. The Oracle SQL Optimizers. Recuperado de http://www.remote-dba.net/t_tuning_sql_optimizers.htm
[36] Burleson, D. Oracle SQL tuning Goals. Recuperado de http://www.remote-dba.net/oracle_10g_tuning/t_sql_tuning_goals.htm
[37] Kyte, T. (2003, Julio/Agosto). On Procedures, Flushes, and Writes. Oracle Magazine. Recuperado de http://www.oracle.com/technology/oramag/oracle/03-jul/o43asktom.html
[38] Gogala, M. Manipulating the CBO in an OLTP Environment. Recuperado de http://www.dba-oracle.com/t_gogala_cbo_oltp1.htm
[39] Kyte, T. (2006, Mayo/Junio). On Joins and Query Plans. Oracle Magazine. Recuperado de http://www.oracle.com/technology/oramag/oracle/06-may/o36asktom.html
[40] Green, C. (2002, Octubre). Gathering Optimizer Statistics. Understanding Statistics. *Oracle9i Database Performance Tuning Guide and Reference Release 2 (9.2)*. (p. 2). Estados Unidos: Oracle Corporation.
[41] Vennapusa, P., (2003). Gathering Statistics. Predicate Selectivity. En N. Brozowski (Ed.), *Oracle 9i Database: SQL Tuning Workshop R2*. (p. 25). Estados Unidos: Oracle Corporation.
[42] Knowledge Xpert for PL/SQL 10.0. SQL Optimizers – Overview.
[43] Green, C. (2002, Octubre). Gathering Optimizer Statistics. Generating Statistics. *Oracle9i Database Performance Tuning Guide and Reference Release 2 (9.2)*. (pp. 3-4).
[44] Vennapusa, P., (2003). Gathering Statistics. Generating Table Statistics. En N. Brozowski (Ed.), *Oracle 9i Database: SQL Tuning Workshop R2.* (pp. 11-20). Estados Unidos: Oracle Corporation.
[45] Chan, I. (2008, Marzo). Managing Optimizer Statistics. Automatic Statistics Gathering. *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*. (pp. 2-3). Estados Unidos: Oracle.
[46] Chan, I. (2008, Marzo). Managing Optimizer Statistics. Manual Statistics Gathering. *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*. (p. 7). Estados Unidos: Oracle.
[47] Green, C. (2002, Octubre). Gathering Optimizer Statistics. Using Histograms. *Oracle9i Database Performance Tuning Guide and Reference Release 2 (9.2)*. (pp. 20-21). Estados Unidos: Oracle Corporation.
[48] Vennapusa, P., (2003). Gathering Statistics. When to Use Histograms. En N. Brozowski (Ed.), *Oracle 9i Database: SQL Tuning Workshop R2.* (p. 32). Estados Unidos: Oracle Corporation.
[49] Chan, I. (2008, Marzo). Managing Optimizer Statistics. Manual Statistics Gathering. Column Statistics and Histograms. Oracle Database Performance Tuning Guide 10g Release 2 (10.2). (p. 7). Estados Unidos: Oracle.
[50] Chan, I. (2008, Marzo). SQL Tuning Overview. Goals for Tuning. *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*. (pp. 1-2). Estados Unidos: Oracle.

[51] Chan, I. (2008, Marzo). SQL Tuning Overview. Identifying High-Load SQL. *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*. (pp. 2-4). Estados Unidos: Oracle.

[52] Chan, I. (2008, Marzo). SQL Tuning Overview. Identifying High-Load SQL. Gathering Data on the SQL Identified. *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*. (p. 4). Estados Unidos: Oracle.

[53] Vennapusa, P., (2003). Following a Tuning Methodology. Overview of SQL Statement Tuning. En N. Brozowski (Ed.), *Oracle 9i Database: SQL Tuning Workshop R2*. (pp. 17-25). Estados Unidos: Oracle Corporation.

[54] Ziauddin, M., Zait, M., Minhas, M., Yagoub, K. (2006, Junio). Automatic SQL Tuning. *The Self-Managing Database: Guided Application & SQL Tuning with Oracle Database 10g Release 2 [An Oracle White Paper]*. (pp. 6-8). Estados Unidos: Oracle Corporation.

[55] Ziauddin, M., Zait, M., Minhas, M., Yagoub, K. (2006, Junio). DBMS_SQLTUNE Package. *The Self-Managing Database: Guided Application & SQL Tuning with Oracle Database 10g Release 2 [An Oracle White Paper].* (pp. 19-21). Estados Unidos: Oracle Corporation.

[56] Chan, I. Oracle Corporation. (2008, Marzo). Automatic SQL Tuning. SQL Tuning Advisor. Advisor Output. *Oracle Database Performance Tuning Guide 10g Release 2 (10.2)*. (p. 12). Estados Unidos: Oracle.

[57] Cyran, M., Lane, P., Polk, J.P. (2005, Octubre). Memory Architecture. Overview of the System Global Area. Allocation and Reuse of Memory in the Shared Pool. *Oracle Database Concepts 10g Release 2 (10.2)*. (p. 12). Estados Unidos: Oracle.