IMPLEMENTACIÓN DE HERRAMIENTAS PARA OPTIMIZAR CONSULTAS DE ACCESO A LA INFORMACIÓN DE BASES DE DATOS RELACIONALES

Ing. Wilbert Manzanilla Vidal[1], Dr. Severino Feliciano Morales[2], M. en C. Valentín Alvarez Hilario [3], M.C. Edgardo Solís Carmona [4] y M. en C. Felix Molina Ángel[5]

***Resumen*— El acceso a las bases de datos que albergan grandes volúmenes de información, puede implicar un considerable uso de recursos de procesamiento al momento de hacer consultas. En algunos casos, el tiempo requerido para generar la totalidad de registros recuperados afecta al desempeño de la aplicación al procesar las solicitudes, específicamente cuando la codificación de las Queries no es la adecuada. Para afrontar esta situación, existen aplicaciones que incluyen optimizadores de consultas, los cuales a través de procesos analíticos que aplican heurísticas, permiten determinar la mejor ruta de acceso a los datos, optimizando así los tiempos de respuesta. Estas herramientas son de gran ayuda, debido a que proporcionan información del plan de ejecución de una sentencia SQL, identificando las mejoras que pueden aplicarse.**

En este artículo se pretende demostrar la importancia de la implementación de este tipo de herramientas mediante un ejemplo en una base de datos real para constatar sus ventajas. Para ello haremos uso del software PL/SQL Developer.

***Palabras clave*— Bases de Datos, SQL, queries, optimización.**

Introducción

En las bases de datos relacionales se utilizan sentencias SQL para realizar diversos tipos de operaciones. Desde la creación o modificación de la estructura de una base de datos a través de los comandos DDL (Data Definition Language), la definición de permisos o privilegios de acceso a los objetos mediante comandos DCL (Data Control Language), hasta la recuperación de la información almacenada en diferentes tipos de objetos, los más usados son tablas aunque también se puede acceder a información contenida en vistas, para lo cual se hace uso de comandos DML (Data Manipulation Language).

Extraer información ordenada y con criterios de filtrado, es una de las actividades más recurrentes. Y se puede realizar mediante sentencias SQL que pueden ser codificadas de formas diversas, cada una con desempeños similares o muy diferentes, dependiendo de varios factores. Uno de los principales radica en la forma en que se haya conceptualizado la estructura de la base de datos, ya que esto tendrá un impacto de consideración en los tiempos de respuesta al ejecutar consultas. La implementación de particiones en las tablas, así como la indexación de campos, en la mayoría de los casos, mejora el acceso a la información y en consecuencia contribuye a minimizar los tiempos en que los datos son recuperados.

Otro factor que sin duda tiene gran relevancia, es el conocimiento que el personal encargado de generar las consultas tiene de la estructura de la base de datos. Es decir, si tiene conocimiento acerca de los campos que permiten activar automáticamente al ejecutar las consultas, los índices de los campos en las tablas a las que se está accediendo. Existen circunstancias, particularmente cuando se incorporan nuevos elementos al equipo de desarrollo, el desconocimiento de la estructura de la base de datos, les impide aprovechar el potencial de las características con las que fue creada, incluso a generar consultas que podrían comprometer el funcionamiento de la base de datos o de las aplicaciones que la acceden.

Como se ha comentado, las instrucciones DML trabajan sobre los datos almacenados, permitiendo realizar consultas o modificaciones sobre ellos. Las operaciones básicas de manipulación de datos que podemos realizar con SQL se denominan CRUD (Create, Read, Update and Delete). En este artículo nos enfocaremos principalmente a la sentencia SELECT que corresponde a la lectura (Read) de datos, ya que en ella se especifican los campos que se

[1] El Ing. Wilbert Manzanilla Vidal es Responsable Tecnológico del Sistema de Recursos Humanos en la Universidad Autónoma de Guerrero, México. wilmanvi@hotmail.com (autor corresponsal).
[2] El Dr. Severino Feliciano Morales es Profesor de Programación y Desarrollo de Software en la Universidad Autónoma de Guerrero, México y es Dr. en Informática por la Universidad de Murcia, España. sevefelici@hotmail.com
[3] El M. en C. Valentín Alvarez Hilario es Profesor de Base de Datos en la Universidad Autónoma de Guerrero, México. valentin_ah@yahoo.com
[4] El M. en C. Edgardo Solís Carmona es Profesor de Base de Datos en la Universidad Autónoma de Guerrero, México. esoliscr@hotmail.com
[5] El M. en C. Félix Molina Ángel es Profesor de Redes en la Universidad Autónoma de Guerrero, México. molina@uagro.mx

desean consultar, así como los criterios para su selección.

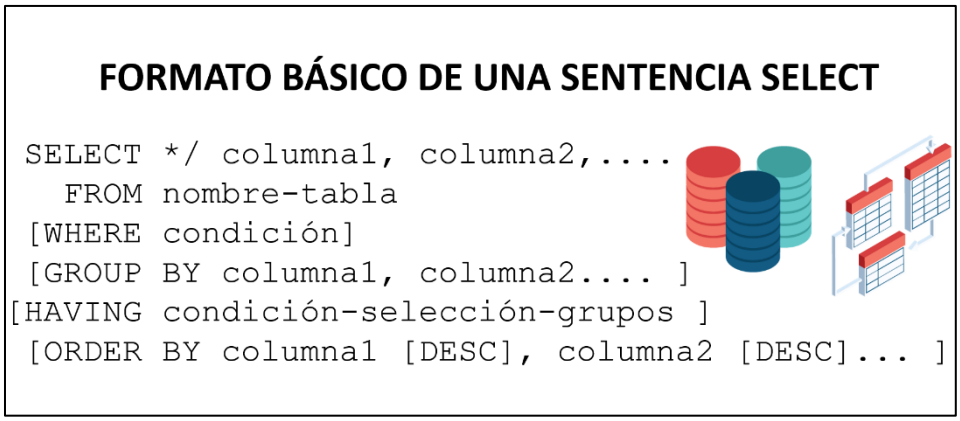

Figura 1. Formato básico de una sentencia SELECT

La eficiencia en la ejecución de una consulta, está estrechamente relacionada con la forma en que se codifique la sentencia SELECT. Para ello, es importante identificar su estructura (figura 1) y la función que cada uno de sus componentes realiza. Es importante mencionar que aunque los únicos elementos requeridos en la codificación de una sentencia de consulta son SELECT y FROM, es altamente recomendable que puedan incluirse los criterios de selección (WHERE) y en caso de requerir datos agrupados por criterios (GROUP BY), definir para estos las condicionales de grupo que aplicarán (HAVING). De igual manera, se deberá cuidar cómo se establecen los criterios para mostrar los resultados (ORDER BY) ya que esto invariablemente requiere de un esfuerzo adicional que tiene un impacto en los tiempos de procesamiento y ejecución.

Alternativas para la optimización de consultas

La optimización de consultas se puede realizar de forma automática (a través de aplicaciones que incorporan esa funcionalidad) o manual. La implementación de herramientas de software que ayudan a identificar aquellas sentencias que representan, o pueden representar, alguna complejidad al momento de su ejecución, comparten el mismo objetivo: reducir el tiempo de respuesta del usuario, lo que significa disminuir el tiempo entre el momento en que un usuario emite una sentencia y recibe una respuesta, así como mejorar el rendimiento, lo que significa usar la menor cantidad de recursos necesarios para procesar todas las filas a las que accede una sentencia. Cuando se opta por realizar la optimización de forma manual, es indispensable contar con sólidos conocimientos. Los principales se muestran en el cuadro 1.

Conocimiento requerido	Descripción
Arquitectura de la base de datos	La arquitectura de la base de datos no debe ser dominio solo del administrador. Como desarrollador, se desea desarrollar aplicaciones en el menor tiempo posible, lo que requiere la explotación de la arquitectura y las características de la base de datos.
SQL y PL/SQL	Debido a la existencia de herramientas basadas en interfaces gráficas de usuario (GUI), es posible crear aplicaciones y administrar una base de datos sin saber SQL. Sin embargo, es imposible optimizar las aplicaciones o una base de datos sin conocer SQL.
Herramientas de optimización SQL	La base de datos, por sí misma, genera estadísticas de rendimiento y proporciona herramientas de optimización de SQL que interpretan estas estadísticas.

Cuadro 1. Conocimientos Requeridos para la optimización de consultas.

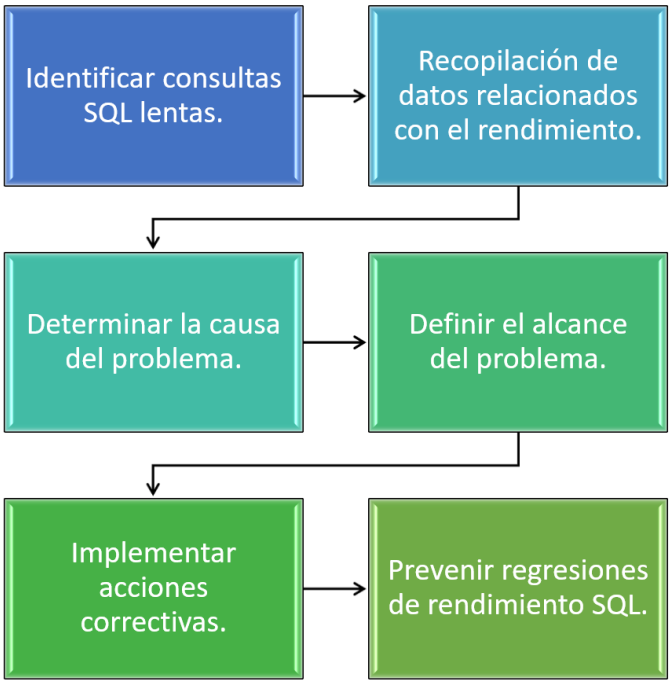

Figura 2. Actividades para la optimización de consultas.

En la optimización de consultas se llevan a cabo todas o algunas de las siguientes actividades mostradas en la figura 2. La primera consiste en identificar las consultas lentas; este proceso se puede realizar haciendo consultas a los registros que la base de datos almacena de cada una de las consultas ejecutadas. El rol de los usuarios críticos es muy importante, ya que ellos también manifiestan tiempos excesivos de espera al recuperar información. Una vez que se han identificado las consultas que se ejecutan con lentitud, se debe recopilar la información del rendimiento asociado a dichas consultas como la estructura de las tablas o vistas a las que la sentencia accede, así como la definición de cualquier índice disponible para dicha sentencia.

El siguiente paso consiste en determinar qué factor está causando el bajo rendimiento de la consulta. Uno de los principales tiene que ver con una codificación deficiente. El más común, es la falta de la definición de la relación entre dos objetos, lo que da como resultado, la generación de un producto cartesiano, como se aprecia en la figura 3.

AUTORES	
EMP_ID	EMP_NOMBRE
1	SEVERINO
2	ALVAREZ
3	SOLIS
4	MOLINA

ARTICULOS	
ART_ID	ART_DESCRIP
101	METADATOS
201	OPTIMIZACION SQL
301	REDES IPV6

Producto cartesiano (select * from autores, artículos)			
EMP_ID	EMP_NOMBRE	ART_ID	ART_DESCRIP
1	SEVERINO	101	METADATOS
1	SEVERINO	201	OPTIMIZACION SQL
1	SEVERINO	301	REDES IPV6
2	ALVAREZ	101	METADATOS
2	ALVAREZ	201	OPTIMIZACION SQL
2	ALVAREZ	301	REDES IPV6
3	SOLIS	101	METADATOS
3	SOLIS	201	OPTIMIZACION SQL
3	SOLIS	301	REDES IPV6
4	MOLINA	101	METADATOS
4	MOLINA	201	OPTIMIZACION SQL
4	MOLINA	301	REDES IPV6

Figura 3. Ejemplo de producto cartesiano.

Si bien los errores humanos son la primera causa de un mal desempeño de una consulta SQL, también hay otros factores que pueden incidir en la ejecución lenta de las consultas, por ejemplo, si las estadísticas de la base de datos no están actualizadas, podría provocar que al momento de elegir un plan de ejecución no se opte por el mejor, debido a que las estadísticas que se consultan son obsoletas. Un mal funcionamiento en los componentes de hardware, también pueden provocar lentitud en el desempeño de una consulta.

Una vez que se ha identificado la causa del problema, se debe definir el alcance del mismo, es decir, si el problema radica en una codificación deficiente, bastará con considerar su corrección. Pero si se identifica una configuración limitada en los parámetros de la base de datos, en algunos casos, cambiar esos parámetros pueden requerir de un reinicio de la base de datos, lo que directamente afectará a todas las aplicaciones que utilizan la información contenida en ella. Este tipo de mantenimientos, requieren un trabajo de programación.

La siguiente fase consiste en implementar las acciones correctivas según correspondan a la causa del problema. Cuando se trata de codificaciones deficientes de SQL, se deben hacer las correcciones necesarias al código para que sea más eficiente. Es importante considerar que en ocasiones, las sentencias SQL están escritas de manera correcta, pero contienen características que implican un mayor consumo de recursos, por ejemplo, el uso de funciones en las cláusulas WHERE. De igual manera es altamente recomendable seccionar una sentencia SQL compleja en múltiples sentencias simples. Finalmente, para asegurar un desempeño SQL óptimo, se debe verificar que el plan de ejecución provee el óptimo funcionamiento y que se ha elegido el mejor plan disponible.

Fases del procesamiento de una sentencia SQL

Durante el procesamiento de una sentencia SQL se distinguen cuatro fases las cuales se muestran en la figura 4. En la primera fase (analizador) se asegura que la consulta sea sintácticamente y semánticamente correcta, es decir, que esté correctamente escrita (por ejemplo, sin comas o paréntesis fuera de lugar) y que los objetos referenciados en la sentencia existan. En caso de que sea incorrecta notifica los errores encontrados. De lo contrario, convierte la sentencia en una expresión algebraica y la pasa a la siguiente fase que es la encargada de realizar optimizaciones directas, considerando diferentes planes de consulta, eligiendo el plan óptimo.

La tercera fase, recibe el plan de ejecución óptimo del optimizador y produce un plan de ejecución iterativo que puede ser utilizado por el resto de la base de datos. El plan iterativo es un programa binario que, cuando es ejecutado por el motor SQL, produce un conjunto de registros. La fase final es la ejecución, durante la cual, el motor SQL ejecuta cada registro en el árbol producido por el generador de registros. Este paso es el único obligatorio en el procesamiento de DML. Y conlleva a la obtención de los resultados.

Figura 4. Fases del procesamiento SQL

Como se ha podido notar, el plan de ejecución contiene información relevante que será de utilidad al momento de realizar optimizaciones a una sentencia SQL, ya que describe un método de ejecución recomendado para una sentencia y muestra la combinación de los pasos que utiliza la base de datos para ejecutarla. Cada paso recupera registros de datos físicamente de la base de datos o los prepara para el usuario que ejecuta la sentencia. El plan de ejecución muestra el costo de todo el plan, así como de cada operación por separado.

Caso práctico

Considerando dos catálogos; uno de Estados de la República (almacenado en la tabla TCATESTADOS) que contiene 34 registros y otro de Localidades (almacenado en una tabla llamada TCATLOCALIDADES) la cual contiene 2576 registros, que aunque no son muchos, servirá para ejemplificar cómo se procesa una sentencia SQL y cómo implementar mejoras en la sentencia que disminuya los tiempos de respuesta y optimice el uso de recursos. Para ello utilizaremos la funcionalidad "Explain plan" del software PL/SQL Developer. Si se desea consultar todos los campos de los registros almacenados en las tablas se observará el resultado en la figura 5. En ella se aprecia la codificación de la sentencia SQL y el plan de ejecución que el optimizador ha devuelto como mejor opción. Como se ha prescindido de especificar criterios de selección (al no contener la condición WHERE) el plan de ejecución identifica un acceso completo a las tablas, por lo que como se aprecia en la imagen, se obtiene un producto cartesiano con una cardinalidad igual al producto de los registros contenidos en ambas tablas (2576*34=87584). Adicionalmente se muestra el costo de la sentencia y el número de bytes requeridos.

```
select *
  from TCATLOCALIDADES L, TCATESTADOS E
```

Objetivo del optimizador: All rows

Description	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL ROWS		98	87584	8495648
MERGE JOIN CARTESIAN		98	87584	8495648
TABLE ACCESS FULL	TCATESTADOS	2	34	1632
BUFFER SORT		96	2576	126224
TABLE ACCESS FULL	TCATLOCALIDADES	3	2576	126224

Figura 5. Acceso completo a las tablas TCATLOCALIDADES y TCATESTADOS

A continuación aplicaremos algunas adecuaciones a la sentencia. El primero consistirá en establecer la unión de igualdad entre los campos que relacionan ambos objetos (LOC_ESTADODO y EDO_ID). Esta simple acción reduce de manera considerable todos los valores de la sentencia que la precedió. La segunda adecuación consistirá en especificar un criterio de selección, para ello se añadirá a la cláusula WHERE la condición de que solo se obtengan los registros de localidades que pertenezcan al Estado de Guerrero. Como se podrá observar en la figura 6, cada vez que se aplicó una modificación a la sentencia, hubo una variación significativa en el plan de ejecución que el optimizador eligió como óptimo. En relación a la sentencia original (ilustrada en la figura 5), cuando se estableció la relación entre las tablas, el valor de la cardinalidad disminuyó al valor de la tabla que contiene más registros (TCATLOCALIDADES). Y cuando se especificaron los criterios de selección en la cláusula WHERE, estos valores descendieron aún más hasta coincidir con el número de localidades que están registradas para el Estado de Guerrero.

```
select * from TCATLOCALIDADES L, TCATESTADOS E where L.LOC_ESTADO=E.EDO_ID
```

Objetivo del optimizador: All rows

Description	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL ROWS		7	2576	249872
HASH JOIN		7	2576	249872
TABLE ACCESS FULL	TCATESTADOS	2	34	1632
TABLE ACCESS FULL	TCATLOCALIDADES	4	2576	126224

```
select E.ESTADO, L.LOC_LOCALIDAD from TCATLOCALIDADES L, TCATESTADOS E
 where L.LOC_ESTADO=E.EDO_ID and E.ESTADO='GUERRERO'
```

Objetivo del optimizador: All rows

Description	Object name	Cost	Cardinality	Bytes
SELECT STATEMENT, GOAL = ALL ROWS		7	76	5092
HASH JOIN		7	76	5092
TABLE ACCESS FULL	TCATESTADOS	2	1	31
TABLE ACCESS FULL	TCATLOCALIDADES	4	2576	92736

Figura 6. Adecuaciones a la consulta.

Es importante observar que a pesar de que se ha especificado la relación de unión entre los dos objetos y se han establecido los criterios de selección, el plan de ejecución sigue considerando accesos completos a las tablas. Lo anterior, deja en evidencia que dichos objetos no contienen índices que ayuden a optimizar la consulta. Por lo que procederemos a la implementación de los mismos y observaremos los resultados en la figura 7.

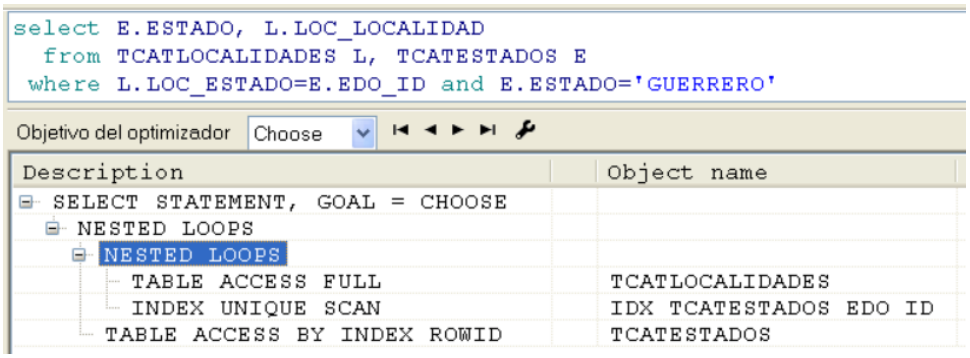

Figura 7. Implementación de índices para optimización de consulta.

En el plan de ejecución se puede apreciar que si bien se está haciendo una lectura completa de la tabla TCATLOCALIDADES, en el caso de la tabla TCATESTADOS solo se están consultado los registros con ayuda del índice habilitado para el campo EDO_ID. La figura 8, muestra las estructuras de dichos objetos.

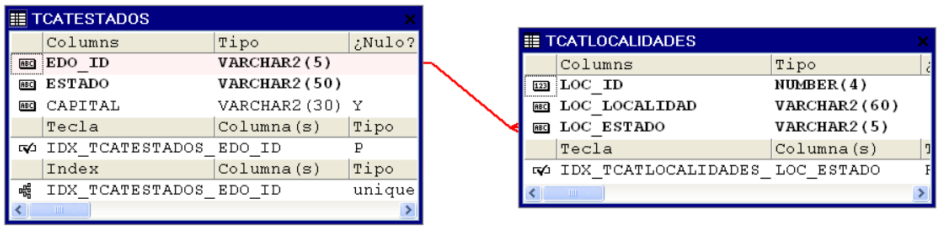

Figura 8. Estructura de las tablas

Conclusiones

Como se ha podido observar, el proceso para la optimización de consultas va más allá de la correcta codificación de una sentencia, es decir, no basta con que la sentencia pueda ser interpretada, sino que ésta pueda especificar con precisión los campos a consultar, los objetos que serán consultados (tablas, vistas, otras consultas), la relación entre estos objetos y particularmente precisar los criterios de selección para evitar en lo posible realizar accesos completos a los objetos involucrados. Se requiere también que al definir las estructuras de los objetos, se considere la necesidad de implementar índices, que sin abusar de ellos, permitan realizar consultas ágiles a los datos que contengan. Una tarea de los administradores de bases de datos, es identificar aquellas tablas que por el volumen de registros que contendrán, requieren de un particionamiento que eficientice el acceso a los mismos.

Es de importancia mencionar que la optimización de consultas debe ser una actividad constante. Y que se debe considerar el hecho de que algunos cambios en las versiones de bases de datos o del hardware que alberga la base de datos, pueden tener un impacto en el desempeño de una consulta. Existen también recomendaciones al momento de la generación de consultas, entre las que podemos destacar; sustituir el uso del operador “or” por la unión de dos consultas simples con los criterios específicos de cada una, la segmentación de consultas complejas por dos o más consultas simples, evitar en lo posible la inclusión de la cláusula “in”, así como limitar el acceso a tablas remotas.

Si bien todo lo descrito en el presente artículo puede realizarse a través de comandos en la consola SQL nativa de la base de datos, implementar herramientas gráficas, facilitan la compresión a través de esquemas visuales que de manera rápida e intuitiva nos permiten identificar las posibles mejoras a implementar.

Referencias

Lance Ashdown (Marzo 2018) Oracle® Database SQL Tuning Guide, ebook. Consultado por Internet el 18 de septiembre del 2018. Dirección de internet: https://docs.oracle.com/en/database/oracle/oracle-database/12.2/tgsql/sql-tuning-guide.pdf

Lance Ashdown, Tom Kyte. (Mayo 2015). Oracle Database Concepts, 11g Release 2 (11.2), evook E40540-04. Consultado por Internet el 21 de septiembre del 2018. Dirección de internet: https://docs.oracle.com/cd/E11882_01/server.112/e40540.pdf

Steve Fogel, . (Mayo 2015). Oracle Database Administrator's Guide, 11g Release 2 (11.2), evook E25494-07. Consultado por Internet el 22 de septiembre del 2018. Dirección de internet: https://docs.oracle.com/cd/E11882_01/server.112/e25494.pdf