TEMA 36

LA MANIPULACIÓN DE DATOS. OPERACIONES. LENGUAJES. OPTIMIZACIÓN DE CONSULTAS

ÍNDICE

1. INTRODUCCIÓN

El lenguaje de descripción de datos es el medio adecuado para declarar al Sistema de Gestión de Base de Datos qué estructuras de datos van a utilizarse.

IBM lanzó el SQL/DS, el primer gestor de Base de Datos Relacional comercial, que corría bajo DOS/VSE y VM/CMS.

Desde entonces el mercado de mainframes, miniordenadores y microordenadores, se ha visto inundado por los sistemas gestores de Bases de Datos que proclaman ser relacionales.

Pero la mayoría de los productos no cumplen completamente el modelo relacional que fue establecido por E. Codd en junio de 1970 en un artículo titulado "A relacional Model of Data for large shared Data Banks", publicado por la revista: Communications of the ACM.

Edgar F. Codd fue el que formuló el concepto de modelo de datos, que se usa para describir las diferentes formas en que los datos se pueden organizar y manipular en un sistema de gestión de Base de Datos. Sin embargo, existen implementaciones anteriores de modelos jerárquicos y de red anteriores a 1970.

Los modelos que soportan estos sistemas no fueron definidos totalmente hasta mucho después por abstracción de los productos existentes. Los modelos relacionales, por el contrario, no aparecieron hasta que se definió el modelo relacional.

El trabajo de Codd produjo un gran movimiento de investigadores y experimentos, incluyendo el desarrollo de prototipos de unos cuantos lenguajes relacionales que materializaban algunas o todas las características del modelo relacional. Uno de ellos era el lenguaje llamado Structured English Query Language, o SEQUEL, que fue definido por D.D. Chamberlin y otros, en los laboratorios de investigación de IBM en San José, California, en 1974. Una versión llamada SEQUEL/2 fue definida en 1976 y más tarde se llamó SQL.

Todas las Bases de Datos tienen dos partes fundamentales: una la parte Estática compuesta por los objetos, las interrelaciones, las restricciones de integridad y la forma de representación. Y por otro lado esta la parte Dinámica que engloba todas las operaciones de recuperación, de actualización y la forma de realizarlas (es decir, los tipos de actuación)

Para estas dos partes fundamentales existen dos tipos de lenguajes que son: LDD (Lenguaje de Descripción de Datos) para la primera parte y el LMD (Lenguaje de Manipulación de Datos) para la segunda.

Existen diferentes tipos de lenguajes, podemos agruparlos de la siguiente manera:

- Huésped: Necesita de un lenguaje anfitrión para poder funcionar. Como ejemplo el Cobol con Bases de Datos.

- Autocontenido: Funciona de manera autónoma, para trabajar con Bases de Datos. Como ejemplo el SOL.

- Procedimental y no procedimental: En este último no se pueden definir procedimientos ni funciones.

- Diferido (Batch) y conversacional.

- Navegacional: (Registro a registro) Encuentra un elemento y luego sigue buscando a ver si encuentra otro que cumpla la misma condición y así sucesivamente.

- Especificación: (Conjunto de registros) Va a todos los elementos.

2. LENGUAJES DE MANIPULACIÓN DE DATOS

Para cumplir los objetivos asignados a la función de manipulación se ha de contar con lenguajes que den a los usuarios la posibilidad de referirse a determinados conjuntos de datos que cumplan ciertas condiciones (criterio de selección), como que un atributo tenga un determinado valor, o que un conjunto de atributos y valores satisfagan cierta expresión lógica. Además deL criterio de selección, es preciso indicar la estructura externa que se desea actualizar o recuperar.

Una vez especificados el criterio de selección y los datos a actualizar o recuperar, el SGBD debe ocuparse de acceder al correspondiente soporte físico de donde se extraerán los datos definidos para su impresión o trasferencia a un dispositivo de salida, o en donde se insertarán, modificarán o borrarán los datos si se trata de una actualización.

Un programa escrito en un lenguaje anfitrión que interactúa con una Base de Datos puede tener la siguiente estructura:

.....
SENTENCIAS DECLARATIVAS
.....
.....
LLAMADA A LA VISTA DE USUARIO (interacción con la Base de Datos)
.....
declaración de otras áreas de E/S
declaración de áreas para la comunicación de mensajes
.....
.....
SENTENCIAS DE PROCESO
.....
.....
LLAMADA A LA SENTENCIA DEL LMD (interacción con la Base de Datos)
comprobación del contenido del área de mensajes
.....
.....

En la parte declarativa se hace una llamada a la vista de usuario; además, se definirán las áreas de entrada/salida y un área donde pasarán los mensajes enviados por el SGBD para informar sobre distintos aspectos relativos a la ejecución de una sentencia.

En la parte de proceso se hace una llamada a la Base de Datos mediante las sentencias de recuperación o actualización del LMD que transfiere los datos desde la Base de Datos al programa si se trata de una sentencia de recuperación, o en sentido inverso si la sentencia es de actualización. En ambos casos habrá que comprobar el contenido del área de mensajes, a fin de saber si la correspondiente sentencia ha finalizado con éxito o no.

El programador necesita un lenguaje de manipulación, al igual que el usuario no informático que deberá disponer de un instrumento análogo para comunicarse con la Base de Datos y extraer de ella (o introducir) las informaciones que precise. Para atender a estas necesidades los SGBD cuentan con lenguajes autocontenidos que dan facilidades a los usuarios con pocos conocimientos de programación para acceder a la base y manipular de forma sencilla los datos almacenados en ella, sin necesidad de apoyarse en ningún lenguaje de programación y, en general en modo interactivo desde un terminal. Los lenguajes autocontenidos suelen incluir no sólo los medios de manipulación, sino también facilidades para describir los datos que se desea recuperar o actualizar.

A veces, el mismo lenguaje de manipulación puede actuar como huésped y como autocontenido, así ocurre, por ejemplo, con el SQL, que puede ser llamado desde un lenguaje anfitrión (como Pascal) o bien puede interactuar directamente con la Base de Datos sin necesidad de estar apoyado en ningún lenguaje de programación. Codd (1982) se refiere a esta clase de lenguajes, diciendo que gozan de la propiedad relacional uniforme, aunque posteriormente se ha referido a ella con el nombre de propiedad dual (Codd, 1990). La posibilidad de que un mismo lenguaje de datos pueda usarse, como autocontenido (interactivamente desde un terminal) y también estar incluido en un programa de aplicación, tiene claras

ventajas para los usuarios en cuanto a facilidades de depuración, mejora de la intercomunicación entre analistas, programadores y usuarios finales, evita tener que aprender más de un lenguaje, etc. En definitiva, puede aumentar la productividad de los sistemas que dispongan de tal clase de lenguaje.

Los lenguajes de manipulación de datos, admiten, además, otro tipo de clasificaciones. Un LMD puede ser procedimental o no procedimental. Un lenguaje de manipulación de datos es tanto más procedimental cuanto con más detalle sea preciso especificar el procedimiento necesario para acceder a la Base de Datos a fin de recuperar o actualizar los datos. Los lenguajes orientados al usuario final deben ser lo menos procedimentales posible, mientras que los lenguajes que utilizan los programadores suelen ser bastante más procedimentales. En un lenguaje procedimental basta con decir qué se quiere, sin explicar cómo obtenerlo; mientras que si el lenguaje es más procedimental no es suficiente con indicar el qué, sino que es necesario, además, precisar el algoritmo.

Existen lenguajes de manipulación de datos llamados navegacionales que recuperan o actualizan los datos registro a registro, debiendo el programador indicar el camino que se ha de recorrer, a través de la estructura definida, hasta llegar al registro buscado; en este caso, cada sentencia o llamada al lenguaje de manipulación de datos produce la recuperación o actualización de un registro único (Lenguajes Codasyl o jerárquico). En cambio, otros lenguajes actúan sobre el conjunto de registros, de forma que una única sentencia puede dar lugar a la recuperación o actualización del conjunto de registros que cumpla el criterio de selección especificado (como ocurre con los lenguajes relacionales, por ejemplo SQL).

En general, los lenguajes de tipo huésped son procedimentales, se explotan en diferido y actúan registro a registro, mientras que los autocontenidos suelen ser no procedimentales, se usan en conversacional y recuperan conjuntos de registros. Por ejemplo, el lenguaje DL/I (IBM) es de tipo huésped, procedimental, diferido y navegacional; el lenguaje SQL es no procedimental, actúa sobre el conjunto de registros y puede usarse en modo autocontenido o como lenguaje huésped desde un lenguaje anfitrión, ya que goza de la propiedad dual.

En la actualidad, la mayoría de los SGBD disponen también de lenguajes de cuarta generación, en general muy poco procedimentales, que permiten la consulta y actualización de la Base de Datos.

3. LENGUAJES PROPORCIONADOS A LOS USUARIOS

Todos los usuarios de la Base de Datos, disponen de un conjunto de medios incluidos en el SGBD que les permiten interactuar con la Base de Datos.

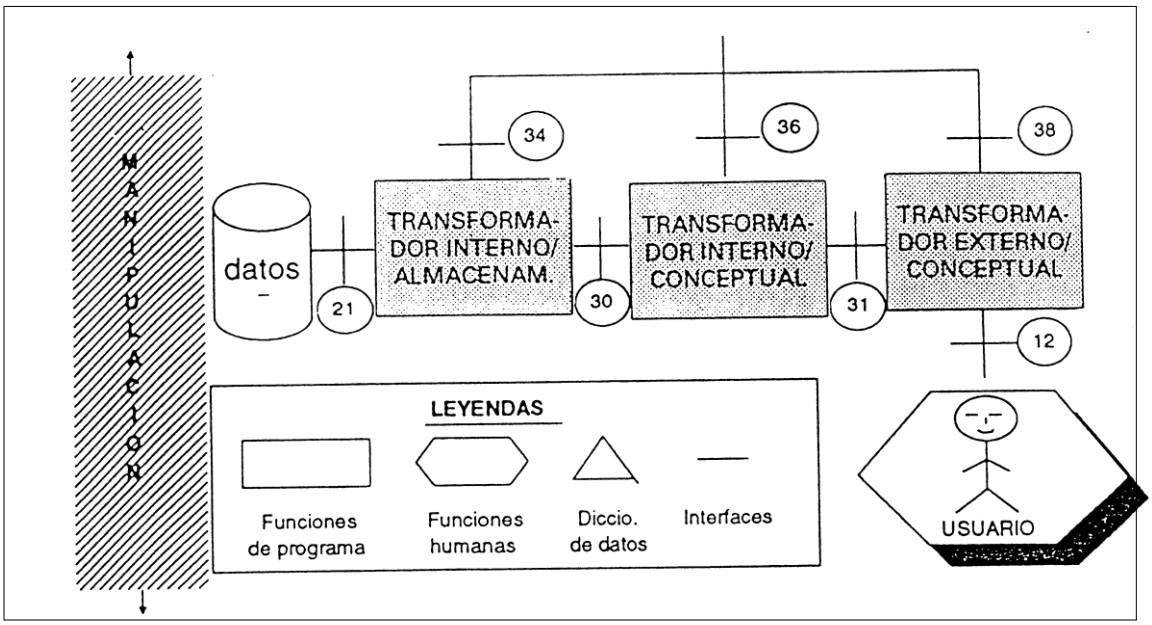
El administrador ha de tener la posibilidad de realizar la definición de los datos a nivel físico y lógico (global y externo), además de disponer de instrumentos que den facilidades para la creación, optimización, reorganización, copias de seguridad, recuperaciones en casos de fallo o caídas del sistema, etc. Estas facilidades se suministran mediante un conjunto de procedimientos que varían considerablemente según el SGBD en concreto.

El usuario informático que interactúa con la base mediante un lenguaje huésped no le incumbe la descripción física de los datos ni la descripción lógica global. En realidad, tampoco debería permitírsele definir sus propias estructuras externas que deben ser responsabilidad sólo del administrador, pero en la práctica, y aunque no es aconsejable, a veces se autoriza a los programadores para que ellos mismos describan las vistas externas correspondientes a las aplicaciones que desarrollan. En lo que se refiere a la función de manipulación, los informáticos suelen disponer de uno o varios lenguajes anfitriones desde donde se hacen las llamadas al lenguaje de Manipulación de datos (LMD), además de generadores de informes, lenguajes de cuarta generación (L4G), y algunas otras ayudas a la programación. Los programadores también pueden utilizar lenguajes autocontenidos, pero no es lo más común, ya que se suele obtener una mayor eficiencia haciendo uso de un lenguaje anfitrión y de las sentencias del lenguaje de Manipulación de datos de tipo huésped o de lenguajes de cuarta generación que incluso pueden provenir de otro suministrador distinto del SGBD.

Los usuarios no informáticos se han dividido en dos grupos, unos cuyas necesidades de información pueden concretarse y formalizarse de antemano, y otros para los que no es posible realizar este proceso de formalización. Para atender a las exigencias de manipulación de los primeros, se suelen preparar procedimientos en los cuales se incluyen facilidades para que el usuario describa la vista externa que

quiere recuperar, o ésta se define en el propio procedimiento, sin dar al usuario opción alguna de modificarla. Cuando el tipo de proceso no es formalizable y/o no se conocen bien de antemano las necesidades de información, el SGBD suele disponer para estos casos de lenguajes autocontenidos que se usan en general en modo conversacional y que incluyen algunas facilidades de descripción.

4. MANIPULACIÓN DE LA BASE DE DATOS



El usuario puede manipular (insertar, borrar, modificar y seleccionar) los datos utilizando la interfaz 12, que podría ser un lenguaje de Manipulación, como SQL o QUEL. Una petición de datos por parte del usuario es ejecutada por los transformadores conceptual/externo, interno/conceptual y almacenamiento/interno, que utilizan los metadatos por medio de las interfaces 38, 36 y 34, respectivamente. La solicitud del usuario en la interfaz 12 la convierten los transformadores en peticiones a las interfaces 31, 30 y 21, que devuelven el resultado al usuario. Estas últimas interfaces constituirán la función de vínculo (Binding) entre los distintos niveles (conceptual, interno y de almacenamiento).

5. OPTIMIZACIÓN DE CONSULTAS

Los SGBD Relacionales ofrecen lenguajes poco procedimentales, ya que el usuario dice el resultado que hay que obtener, pero no tiene que especificar cómo conseguirlo. Esto presenta una gran ventaja para el usuario pero significa que el sistema debe ser capaz de tomar las consultas de alto nivel que se le someten y escoger el camino más apropiado para acceder a los datos requeridos de modo eficiente, usando adecuadamente las tablas, los índices primarios y secundarios, etc., es lo que se suele denominar **navegación automática**.

El System R fue pionero entre los sistemas relacionases, incluía un componente que realizaba esta función: componente que se bautizó con el nombre de **optimizador**. En la actualidad de aplica a cualquier sistema relacional. No se trata de una optimización propiamente dicha, sino de la selección de una estrategia de acceso que responda a unos objetivos de eficiencia global; en algunos productos se le denomina "Planificador de consultas".

Las operaciones relacionales son optimizables precisamente por ser de alto nivel, y un buen optimizador será capaz de reconocer qué es lo que el usuario trata de hacer y responderá de una forma óptima desde el punto de vista de la eficiencia. Por el contrario, en los sistemas que operan registro a registro, las estrategias de acceso son responsabilidad del usuario, habiendo pocas posibilidades de que el sistema pueda optimizar, al estar la estrategia expresada en una secuencia de operaciones de bajo nivel.

Ha sido en los SGBD Relacionales donde la optimización se ha concretado y tratado más formalmente, ya que al ser mayor la independencia Físico/Lógica, la optimización es imprescindible.

Date (1987) ha sugerido, utilizar el criterio de "tener o no optimizador" como una línea divisoria entre los sistemas genuinamente relacionases y aquellos que presentan una apariencia relacional sin aportar todas las ventajas de este modelo. El optimizador es un elemento clave para la eficiencia de un producto.

5.1. Recursos actualmente optimizados

Los procedimientos de optimización tienden a minimizar el tiempo de respuesta. También se podría considerar minimizar el coste total de una consulta a la Base de Datos, en cuyo caso se tienen en cuenta, además del tiempo de respuesta, el ahorro de otros recursos.

El coste total es función, entre otros, de los costes de comunicación, de acceso a almacenamiento secundario (influido por el número de accesos a los datos y a los resultados intermedios), de almacenamiento y de computación.

En estos momentos, los algoritmos de optimización son fijos y el administrador de la Base de Datos no puede marcar sus propios objetivos a este respecto, aunque puede mejorar la eficiencia del sistema, creando, por ejemplo, índices secundarios. Parece ser que en el futuro, especialmente con la introducción de técnicas de Inteligencia Artificial, los optimizadores se flexibilizarán, y el administrador, con su experiencia, podrá realizar ajustes en el sistema de optimización, señalando ciertas prioridades en cuanto a objetivos.

5.2. Optimización de consultas simples

En un sistema centralizado el procedimiento de optimización consta de cuatro fases:

- Llevar la consulta a una representación interna.
- Aplicar a dicha representación una serie de transformaciones lógicas.
- Convertir la consulta transformada en un conjunto de planes de acceso.
- Elegir el más barato de dichos planes.

Los dos primeros pasos tienen un alto grado de independencia respecto a los datos, mientras que los dos últimos tienen una gran dependencia. Para realizar adecuadamente este proceso ha de mantenerse, en la metabase, información general sobre la estructura de la Base de Datos y también información estadística sobre el contenido de ésta. Pero al igual que en muchos problemas de investigación operativo, el coste de obtener y mantener esta información estadística debe compararse con su propio valor.

5.3. Optimización de consultas múltiples

Cuando se realizan múltiples consultas a la Base de Datos, la forma más común es tratar de optimizar cada una por separado; sin embargo, dado un conjunto de consultas y estudiándolo con una visión global, pueden descubrirse subtareas comunes a varias de ellas que de realizarse de una vez evitan accesos o procesos redundantes, mejorándose el tiempo de respuesta.

La descomposición de una consulta en sus componentes básicos es una técnica ya tradicional en la optimización. En el caso de consultas múltiples se puede aplicar la misma técnica, teniendo en cuenta que en esta descomposición es necesario descubrir las subexpresiones comunes a varias consultas. A partir de estas componentes y de sus interrelaciones se puede construir un grafo que represente una consulta múltiple y aplicarle técnicas clásicas de búsqueda en grafos y de minimización de costes.

5.4. Optimización semántica

La optimización semántica utiliza las restricciones de integridad para transformar las consultas. El principio es muy simple, a cada predicado P de la consulta puede añadirse un número arbitrario de restricciones de integridad C_1, C_2, C_3, ... C_n y formarse así un nuevo predicado C_1 AND C_2 AND C_3 ... AND C_n. Esta operación no cambia el resultado de la consulta pues, por definición, todas las restricciones de integridad son siempre ciertas. El nuevo predicado puede ser transformado semánticamente en otro equivalente de forma que su evaluación sea más fácil. A este proceso se le llama "simplificación semántica" si la consulta resultante tiene menos términos o menos variables libres que el predicado original.

El principal objetivo de la optimización semántica es la reducción del espacio de búsqueda por la aplicación de restricciones de integridad y la transformación de la consulta para facilitar su evaluación. El tipo de restricciones de integridad existentes en el sistema influye especialmente en la reducción del espacio de búsqueda.

La optimizaci6n semántica, a pesar de sus ventajas, apenas se aplica todavía en los productos comerciales.

5.5. Optimizadores

Optimizador de consultas (INGRES)

El optimizador de INGRES es estadístico, permite tener una serie de estadísticas de distribución de datos. Dicha información está guardada en la propia Base de Datos en forma de histogramas.

Este optimizador almacena información estadística sobre los valores de una cierta columna de una tabla. De esta manera, cuando el optimizador tiene que ejecutar una consulta, no tiene que "adivinar", sino que simplemente consultará los histogramas (almacenados en los catálogos del sistema), para determinar el número de filas a devolver.

Con esta información estadística, el optimizador puede determinar, por ejemplo, que "los empleados mayores de 100 años" es una condición bastante restrictiva, y que "los empleados mayores de 5 años" es una condición poco restrictiva.

La decisión fundamental en la optimización de un JOIN, es elegir correctamente la unión, clasificación y mezcla de las filas entre las tablas que forman parte de la consulta. Sólo un optimizador como el de INGRES, basado en estadísticas, podrá determinar estas condiciones.

INGRES busca el mejor plan de ejecución, y nos lo puede mostrar por pantalla si lo deseamos. En él veremos cómo ha hecho INGRES las uniones, las estructuras de las tablas, etc.

No será importante el orden en que se escriban las sentencias de SQL, INGRES monta la sentencia de la forma que cree que será más rápido.

Optimizador (ORACLE)

El optimizador ORACLE utiliza una serie de algoritmos complejos, que determinan el camino de acceso óptimo. Para ello, cuenta con la información disponible en el Diccionario de Datos y reglas generales sobre qué es mejor en cada situación. Utiliza un esquema de rangos para cláusulas WHERE.

Optimizador (SYBASE)

Trabaja sólo sobre las funciones de manipulación de datos, comandos tales como "create table" o "create index" no son optimizados.

El optimizador elabora, tras una cuidadosa compilación, el conjunto de pasos necesarios para ejecutar la función propuesta, a este conjunto de pasos se les denomina Query Plan.

Para estar ejecutándose el menor tiempo posible, el optimizador de SYBASE limita el número de caminos que van a ser estudiados.

6. BIBLIOGRAFÍA

Gary W. Hansen
Diseño y Administración de Bases de Datos
Prentice Hall, 1998

Alberto Prieto
Introducción a la Informática
Mc Graw-Hill, 2ª edición, 1997

Alfonso Ureña López
Fundamentos de Informática
Ra-ma, 1997