Preparador Informática

www.preparadorinformatica.com

TEMA 36. INFORMÁTICA

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

TEMA 36 INF: LA MANIPULACIÓN DE DATOS. OPERACIONES. LENGUAJES. OPTIMIZACIÓN DE CONSULTAS

1. INTRODUCCIÓN

Hoy en día el mayor activo de las organizaciones son los datos y su gestión eficaz y segura. Por ello, si analizamos la mayoría de los ámbitos de actividad, nos encontramos que la utilización de las bases de datos está ampliamente extendida (al registrarse en una web, al acudir a la consulta médica, al consultar el catálogo de productos de una tienda online, etc.). Las bases de datos y los datos contenidos en ellas, son imprescindibles para llevar a cabo multitud de acciones.

Una vez definida la base de datos (definición de datos), para poder trabajar con ella hay que realizar operaciones de manipulación de datos. En el presente tema vamos a empezar definiendo y diferenciando los conceptos de base de datos y sistema gestor de base de datos y posteriormente nos adentraremos en la manipulación de datos, sus operaciones y analizaremos el funcionamiento de los lenguajes de manipulación de datos. Para ello, se detallarán las órdenes para consulta, inserción, modificación y borrado de información. Terminaremos el tema explicando el modo de realizar la optimización de consultas.

2. LA MANIPULACIÓN DE DATOS

Podemos definir una **base de datos** como un conjunto, colección o depósito de datos almacenados en un soporte informático no volátil. Los datos están interrelacionados y estructurados.

Un **sistema gestor de base de datos o SGBD** es una colección de programas de aplicación que permite a los usuarios la creación y el mantenimiento de una base de datos, facilitando la definición, construcción y manipulación de la información contenida en ésta.

En la actualidad debido al gran volumen de datos que manejamos se requiere de sistemas gestores de bases de datos robustos, que nos permitan acceder y gestionar los datos de un modo eficaz y eficiente.

Un SGBD desarrolla tres funciones fundamentales como son las funciones de definición, manipulación y control de los datos. Me centraré en cómo se realiza la manipulación de datos que es de lo que este tema trata.

La **manipulación de datos** debe permitir realizar una serie de tareas que los SGBD deben llevar a cabo para gestionar los datos de una BD:

- Consultar determinada información.
- Insertar nuevos datos.
- Modificar los datos existentes.
- Eliminar datos que ya no son necesarios.

3. LENGUAJE DE MANIPULACIÓN DE DATOS (DML).

El lenguaje de manipulación de datos está compuesto por un conjunto de comandos que permiten las operaciones necesarias de manipulación o actualización de base de datos (inserción de datos nuevos, modificación de los datos ya existentes y borrado de datos que ya no sean necesarios) así como la recuperación de datos (consultas).

Los SGBD basados en distintos modelos de datos tienen diferentes DML pero se trata de una gramática completa y, generalmente, fácil de entender por usuarios no expertos. El DML es muy dependiente tanto del modelo de base de datos como del SGBD concreto utilizado. Cada modelo de BD tiene su propia organización y distribución de datos por lo que las operaciones necesarias para acceder a la información son completamente diferentes entre unos y otros.

Por su parte cada SGBD tiene su DML, que puede ser propietario o bien una implementación de un estándar o genérico pero complementado con operaciones propias que aportan nuevas funcionalidades o permiten realizar operaciones de forma más eficiente y por tanto los distinguen de la competencia.

3.1. TIPOS.

Pueden existir distintas clasificaciones:

a) Procedimentales o No Procedimentales:

- **Procedimentales**: En ellos se requiere que, en las sentencias del lenguaje, el usuario (normalmente usuario experto) debe especificar qué datos se necesitan y cómo hay que obtenerlos.
- **No procedimentales**: En ellos sólo se requiere que se especifique qué datos se desean sin decir cómo se deben de obtener.

b) Basados en álgebra o cálculo relacional:

- **Basados en álgebra relacional**: El álgebra relacional constituye un lenguaje de consulta procedimental y cuenta con un conjunto de operaciones básicas para consultar una base de datos. Se utilizan una colección de operadores de alto nivel que, aplicados a las relaciones, dan como resultado nuevas relaciones. Los operadores se dividen en operadores primitivos (selección, proyección, unión, diferencia y producto cartesiano) y operadores derivados (intersección, cociente, reunión y reunión natural). El lenguaje SQL (Structured Query Language) está basado en álgebra relacional.
- **Basados en cálculo relacional**: Se puede subdividir en cálculo relacional de tuplas y cálculo relacional de dominios. Es un lenguaje no procedimental. Los lenguajes QBE (Query By Example) y QUEL (Query Language) están basados en cálculo relacional.

c) Estándar o propio:

- **Estándar**: Los sistemas que implementan estándares pueden incorporar directamente el estándar o variaciones de un estándar.
- **Propio**: Pueden implementar el lenguaje completamente o bien unas pocas operaciones.

3.2. CARACTERÍSTICAS.

Las características más importantes de los DML son:

- Dependencia del SGBD.
- Simplicidad y uniformidad.
- Son lenguajes formales, con base matemática.

4. OPERACIONES

Las operaciones que pueden llevarse a cabo por medio de un DML son la consulta, inserción, modificación y borrado de información. La diferencia entre los distintos DMLs radica en cómo se llevan a cabo estas operaciones. Para ello se han ido desarrollando lenguajes que están basados en el álgebra y en el cálculo de predicados.

La diferencia básica entre un lenguaje basado en el álgebra relacional y otro basado en el cálculo relacional, es que el primero es un lenguaje procedimental y el segundo es no procedimental. Codd demostró que el álgebra y el cálculo relacional son lógicamente equivalentes, lo que implica que toda consulta realizada utilizando el álgebra relacional tiene su equivalente en el cálculo relacional.

4.1. ÁLGEBRA RELACIONAL

El álgebra relacional es una disciplina matemática construida a partir de cinco operadores básicos sobre los que se apoyan una serie de axiomas y teoremas, siendo especialmente importante el que implica su carácter de completitud. Gracias a este resultado, demostrado por Codd, se puede demostrar que cualquier acceso a una base de datos relacional puede realizarse en términos de estos operadores.

Los operadores del álgebra relacional pueden clasificarse en:

- **Operadores primitivos**: Son los operadores esenciales que no pueden obtenerse de otros y gracias a ellos el álgebra relacional es un lenguaje completo.
- **Operadores derivados**: Se pueden obtener aplicando varios de los operadores primitivos. No aportan nada nuevo, pero son muy útiles y simplifican muchas operaciones habituales.

4.1.1. OPERADORES PRIMITIVOS:

a) **Unión de conjuntos**: La unión de dos relaciones R y S, denotada por R U S, es el conjunto de tuplas que pertenece a R, a S ó a ambas. Sólo se aplica este operador a relaciones del mismo grado y definidas sobre los mismos atributos.

b) **Diferencia de conjuntos**: La diferencia de dos relaciones R y S, denotada por R – S, es el conjunto de tuplas de R que no pertenecen a S. Las mismas restricciones del caso anterior se verificarán sobre R y S.

c) **Producto cartesiano**: Sean R y S dos relaciones de grados m y n respectivamente. El producto cartesiano R*S es una relación de grado m+n,

constituida por todas las posibles tuplas, en que los m primeros elementos constituyen una tupla de R, y los n últimos una tupla de S.

d) **Proyección**: La proyección $\pi_x(R)$, donde R es una relación definida sobre J (esquema o conjunto de atributos posibles) y x está incluido o coincide con J, es una relación construida por las columnas de R correspondientes a los atributos de x.

e) **Selección**: Sea F una fórmula que involucra:
   - Operandos constantes, y que referencian a atributos de la relación implicada R.
   - Operadores aritméticos de comparación: <,>,=,<=,>=,<>.
   - Operadores lógicos: AND, OR.

   Entonces la selección $\sigma_F(R)$ es el conjunto de tuplas de R tales que una vez sustituida en la fórmula F, las ocurrencias de los atributos que en ellas se referencian, resulta verdadera.

4.1.2. OPERADORES DERIVADOS:

a) **Intersección**: Es el inverso a la resta, y se resuelve por las tuplas comunes a las dos relaciones que se intersectan

   $R \cap S = (R - (R - S))$

b) **División o cociente**: Sean R y S relaciones de grado r y s respectivamente, donde r>s y S<>∅. Entonces el cociente R:S es el conjunto de tuplas t de grado (r – s) tales que para toda tupla u de S, la tupla t está en R.

c) **Reunión o Join**: Consiste en aplicar el producto cartesiano a dos relaciones y a la relación obtenida aplicarle una selección que elimine las tuplas que no cumplan una determinada condición.

d) **Reunión Natural o NJoin**: Sean R1 y R2 relaciones con alguna columna con dominio común, la reunión natural de R1 y R2 se calcula:
   - Realizar el producto cartesiano de R1 y R2.
   - Para cada atributo Ai de dominio común en R1 y R2, se seleccionan las filas en las que el valor de Ai en R1 coincide con el Ai de R2.
   - Eliminar la o las columnas de R2 utilizadas en la selección.

4.2. CÁLCULO RELACIONAL DE TUPLAS

Una tupla se define como una función finita que asocia unívocamente los nombres de los atributos de una relación con los valores de una instanciación de la misma. En términos simplistas, es una fila de una tabla relacional.

El cálculo relacional basado en tuplas está basado en el cálculo de predicados utilizando variables-tupla que representan tuplas.
Una consulta es de la forma: $\{T \mid \varphi(T)\}$ donde T es una variable tipo tupla y $\varphi(T)$ es una fórmula que describe a T. El resultado de esta consulta es el conjunto de todas las tuplas t para las cuales la fórmula es verdadera.

Una variable tipo tupla T es una variable capaz de tomar cualquier valor tupla que pertenece a una relación (o tabla).

La sintaxis es definida a partir de la lógica de primer orden. Donde la variable a utilizar son de tipo tupla. Una variable es libre en una fórmula (o subfórmula) si la (sub) fórmula no contiene ninguna ocurrencia de cuantificadores que la limiten. En una consulta de la forma: $\{T \mid \varphi(T)\}$, T es la única variable libre.

4.3. CÁLCULO RELACIONAL DE DOMINIOS

Es un lenguaje de consulta formal que permite expresar las consultas a partir de fórmulas bien formadas, donde cada variable se interpreta como variante sobre el dominio del atributo de una relación. Las expresiones utilizadas tienen la forma: $\{< X_1, X_2, \ldots, X_n > \mid P(X_1, X_2, \ldots, X_n)\}$
Dónde:

$X_1, X_2, \ldots, X_n$ son variables de dominio

$P(X_1, X_2, \ldots, X_n)$ es una fórmula.

5. LENGUAJES

Los lenguajes más utilizados en cada modelo de BD son los siguientes:

5.1. MODELO JERÁRQUICO (IMS)

Information Management System de IBM es uno de los sistemas de base de datos más antiguos y utilizados antes de la universalización de los SGBD en red y relacionales. Este lenguaje consta de varias órdenes que están incorporadas en un lenguaje principal Pascal:

- **"get":** Para la recuperación de datos:

```
Get First <tipo de registro>
Where <condición>
      ó
Get Next <tipo de registro>
Where <condición>
```

- **"insert":** Para insertar un nuevo registro:

```
Insert <tipo de registro>
Where <condición>
```

- **"replace"**: Para modificar un registro que ya existe del tipo <tipo de registro>. Se utiliza hold para que el sistema sepa que va a modificar un registro.
- **"delete"**: Para eliminar un registro del tipo <tipo de registro>.

5.2. MODELO EN RED (DBTG)

Este lenguaje consta de varias órdenes que están incorporadas en un lenguaje principal Pascal.

- **"find y get":** Localiza un registro en la base de datos y da valores a los punteros de actualidad adecuados.

```
Find any <tipo de registro> using <campo de registro>
```


La operación **get** copia el registro al que apunta el actual de unidad de ejecución de la base de datos a la plantilla adecuada en el área de trabajo de programa. Pueden existir dos registros con el valor especificado, para ello utilizamos:

```
Find duplicate  <tipo de registro> using <campo de registro>
```

Acceso al primer registro dentro de un conjunto con la operación Find:

```
Find first <tipo de registro> within <tipo de conjunto>
```

Para localizar los demás miembros del conjunto:

```
Find Next <tipo de registro> within <tipo de conjunto>
```

- **"store":** Se utiliza para crear un nuevo registro de un tipo determinado de registro.

```
Store <tipo de registro>
```

- **"modify":** Para modificar un registro existente:

Modify <tipo de registro>

- **"erase":** Para eliminar un registro existente:
  Erase <tipo de registro>

Antes de modificar o eliminar un registro debe buscarse primero el registro a eliminar con la operación find for update.

- **"connect":** Para insertar un nuevo registro en una ocurrencia determinada de <tipo de conjunto>:
  connect <tipo de registro> to <tipo de conjunto>
- **"disconnect":** Para eliminar un registro existente en una ocurrencia determinada de <tipo de conjunto>:
  disconnect <tipo de registro> from <tipo de conjunto>
- **"reconnect":** Para pasar un registro existente de una ocurrencia de un conjunto a otra ocurrencia del conjunto <tipo de conjunto>:
  reconnect <tipo de registro> to <tipo de conjunto>

5.3. MODELO RELACIONAL. SQL COMO DML

En SQL utilizando el DML podremos realizar las operaciones de consulta (recuperación de datos) y por otro lado operaciones de manipulación de datos (inserción, modificación y borrado de datos).

a) Consultas.

Para realizar consultas sobre las tablas de las bases de datos disponemos de la instrucción **SELECT**. Con ella podemos consultar una o varias tablas. Es sin duda el comando más versátil del lenguaje SQL. Existen muchas cláusulas asociadas a la sentencia SELECT (GROUP BY, ORDER, HAVING, UNION). También es una de las instrucciones en la que con más frecuencia los motores de bases de datos incorporan cláusulas adicionales al estándar.

Vamos a empezar viendo las consultas simples, basadas en una sola tabla. El resultado de una consulta SELECT nos devuelve una tabla lógica. Es decir, los resultados son una relación de datos, que tiene filas/registros, con una serie de campos/columnas. Igual que cualquier tabla de la base de datos. Sin embargo, esta tabla está en memoria mientras la utilicemos, y luego se descarta. Cada vez

que ejecutamos la consulta se vuelve a calcular el resultado. La sintaxis básica de una consulta SELECT es la siguiente:

```
SELECT [ ALL / DISTINCT ] [ * ]
[ListaColumnas_Expresiones] AS [Expresion]
FROM Nombre_Tabla_Vista
WHERE Condiciones
ORDER BY ListaColumnas [ ASC / DESC ];
```

donde:

- **ALL/DISTINCT:** ALL es el valor predeterminado, especifica que el conjunto de resultados puede incluir filas duplicadas. Por regla general nunca se utiliza. DISTINCT especifica que el conjunto de resultados sólo puede incluir filas únicas. Es decir, si al realizar una consulta hay registros exactamente iguales que aparecen más de una vez, éstos se eliminan.
- **Nombres de campos:** Se debe especificar una lista de nombres de campos de la tabla que nos interesan y que por tanto queremos devolver. Normalmente habrá más de uno, en cuyo caso separamos cada nombre de los demás mediante comas. Se puede anteponer el nombre de la tabla al nombre de las columnas, utilizando el formato Tabla.Columna. Además de nombres de columnas, en esta lista se pueden poner constantes, expresiones aritméticas, y funciones, para obtener campos calculados de manera dinámica. Si queremos que nos devuelva todos los campos de la tabla utilizamos el comodín "*" (asterisco).
- **AS:** Permite renombrar columnas si lo utilizamos en la cláusula SELECT, o renombrar tablas si lo utilizamos en la cláusula FROM. Es opcional. Con ello podremos crear diversos alias de columnas y tablas.
- **FROM:** Esta cláusula permite indicar las tablas o vistas de las cuales vamos a obtener la información.
- **WHERE:** Especifica la condición de filtro de las filas devueltas. Se utiliza cuando no se desea que se devuelvan todas las filas de una tabla, sino sólo las que cumplen ciertas condiciones. Lo habitual es utilizar esta cláusula en todas o en la mayoría de las consultas.
- **Condiciones:** Son expresiones lógicas a comprobar para la condición de filtro, que tras su resolución devuelven para cada fila TRUE o FALSE, en función de que se cumplan o no. Se puede utilizar cualquier expresión lógica

y en ella utilizar diversos operadores como menor, mayor, igual, distinto, etc y otros como LIKE (para la comparación de un modelo), BETWEEN (para un intervalo de valores) ó IN (para especificar una relación de valores concretos). Por supuesto es posible combinar condiciones simples de los operadores anteriores utilizando los operadores lógicos OR, AND y NOT.

- **ORDER BY:** Define el orden de las filas del conjunto de resultados. Se especifica el campo o campos por los que queremos ordenar los resultados.
- **ASC / DESC:** ASC es el valor predeterminado, especifica que la columna indicada en la cláusula ORDER BY se ordenará de forma ascendente, o sea, de menor a mayor. Si por el contrario se especifica DESC se ordenará de forma descendente (de mayor a menor).
- **GROUP BY:** Mediante esta cláusula pueden construirse grupos de tuplas sobre una lista de atributos.
- **HAVING:** Esta cláusula permite restringir el conjunto de grupos que pueden ser formados por la cláusula GROUP BY.

b) Inserción de datos

Se lleva a cabo por medio de la instrucción **INSERT**. La operación de inserción permite añadir una o varias filas en una tabla. La inserción puede llevarse a cabo indicando manualmente cada uno de los datos de la nueva fila a insertar o bien combinándolo con una consulta, de forma que se insertará en la tabla el resultado de la consulta. Para insertar manualmente:

```
INSERT INTO NombredeTabla (columna1, columna2  … columnaN)
VALUES (valor1, valor2, … , valorN);
```

Con esto conseguiríamos guardar en la columna1 el valor1, en la columna2 el valor2 y así sucesivamente. Para insertar datos de otra tabla usando una consulta:

```
INSERT INTO NombredeTabla (columna1, columna2  … columnaN)
SELECT TablaOrigen.campo1, TablaOrigen.campo2, …, TablaOrigen.campoN
FROM TablaOrigen;
```

La condición SELECT puede incluir la cláusula WHERE para filtrar los registros a copiar.

c) Modificación/Actualización de datos.

Se utiliza **UPDATE**. Se encarga de crear una consulta de actualización que cambia los valores de los campos de una tabla especificada basándose en un criterio específico. La sintaxis es la siguiente:

```
UPDATE NombredeTabla SET columna1=valor1, columna2=valor2, … ,
columnaN=valorN WHERE Criterio;
```

Si en una consulta de actualización suprimimos la cláusula WHERE entonces todos los registros de la tabla serán actualizados.

d) Eliminación de datos.

Se utiliza **DELETE**. Se encarga de crear una sentencia de eliminación que elimina los registros de una o más de las tablas listadas en la cláusula FROM que satisfagan la cláusula WHERE. La sintaxis es la siguiente:

```
DELETE FROM NombredeTabla WHERE Criterio;
```

6. OPTIMIZACIÓN DE CONSULTAS

En los lenguajes procedimentales el usuario es quien debe indicar cómo se van a recuperar los datos buscados, por lo que la optimización pasaría por optimizar el algoritmo diseñado.

En los lenguajes no procedimentales, sin embargo, el usuario dice el resultado que quiere obtener, pero no especifica cómo conseguirlo. Esto supone una gran ventaja para el usuario, pero delega en el sistema la toma de estas decisiones, lo que limita notablemente la capacidad del usuario para asegurar que los datos se recuperar de forma óptima. Existen, sin embargo, varias tareas que se pueden llevar a cabo para ayudar al sistema a tomar las decisiones correctas. Si nos centramos en las consultas para optimizarlas tendremos en cuenta:

- **Diseño de las tablas y elección de tipos:** El buen diseño de tablas y vistas puede afectar mucho a las prestaciones. La correcta elección de tipos de datos redundará en una menor ocupación de espacio y evitará conversiones de tipos de datos innecesarias, lo cual también incrementará el rendimiento.
- **Campos calculados:** Obligan al sistema a realizar numerosas operaciones durante su consulta, lo que puede afectar al rendimiento del sistema. Una forma de evitarlo sería incorporar estos campos a la base de datos, calculándolos en el momento que se insertan, de forma que se evitarían todas estas operaciones al realizar las consultas.

- **Desnormalización:** Para evitar tener que recuperar los datos siempre en medio de un JOIN, una opción es desnormalizar, que consiste en romper la 3FN al insertar atributos de una tabla en otra tabla o crear una tabla como el resultado de la unión de otras tablas.
- **Monitor de consultas:** Permite realizar una traza de todas las operaciones realizadas sobre el SGBD para localizar las operaciones más frecuentes y determinar qué consultas se realizan de forma más habitual sobre la BD, que serán las consultas sobre las que se deberá centrar la tarea de optimización.
- **Planes de ejecución:** Revisar el plan de ejecución de la consulta es comprobar cómo se está accediendo a las tablas. Para mejorarlo tenemos distintas opciones: reescribir la consulta, usar índices, modificar el agrupamiento físico y lógico...
- **Índices:** Lo fundamental es decidir adecuadamente sobre qué columnas crear un índice y qué tipo de índice crear. Se recomienda crear índices en claves primarias y foráneas, no crear índices en tablas de muy pocos datos, tener cuidado de crear índices de columnas con muchos nulos, etc.

7. CONCLUSIÓN

Un SGBD desarrolla tres funciones fundamentales como son las funciones de definición, manipulación y control de los datos. En este tema se ha presentado una visión global de la manipulación de datos. También se han descrito las operaciones que pueden llevarse a cabo por medio de un lenguaje de manipulación de datos (DML) y las diferencias entre los distintos DMLs en cómo llevan a cabo estas operaciones. Además, se han descrito los lenguajes de manipulación de datos, los cuales permiten consultar o actualizar los datos de la base de datos, de acuerdo con las especificaciones y las normas de seguridad dictadas por el administrador. Finalmente se han descrito los aspectos a tener en cuenta en el proceso de optimización de consultas.

8. BIBLIOGRAFÍA

- Date D.J.: **Introducción a los sistemas de bases de datos**. Editorial Addison-Wesley
- De Miguel A,y Piattini M:**Fundamentos y modelos de BBDD**. Edit. Ra-Ma
- Korth H. y Silberschatz: **Fundamentos de bases de datos**. Editorial McGraw-Hill