Preparador Informática

www.preparadorinformatica.com

TEMA 39. INFORMÁTICA

TEMA 38. S.A.I.

LENGUAJES PARA DEFINICIÓN Y MANIPULACIÓN DE DATOS DE SISTEMAS DE BASES DE DATOS RELACIONALES. TIPOS. CARACTERÍSTICAS. LENGUAJE SQL.

TEMA 39 INF/ TEMA 38 SAI: LENGUAJES PARA DEFINICIÓN Y MANIPULACIÓN DE DATOS DE SISTEMAS DE BASES DE DATOS RELACIONALES. TIPOS. CARACTERÍSTICAS. LENGUAJE SQL.

Preparador Informática

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.

Para poder trabajar con bases de datos relacionales, lo primero que hay que hacer es definirlas. Una vez definida la base 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 los lenguajes de definición y manipulación de datos especificando los tipos y características de ellos y terminaremos estudiando el lenguaje SQL que es el lenguaje más extendido e importante para trabajar con bases de datos relacionales. El lenguaje SQL, Structured Query Language, es un estándar implementado por los principales motores o sistemas de gestión de bases de datos relacionales.

Preparador Informática

2. LENGUAJES PARA LA DEFINICIÓN Y MANIPULACIÓN DE DATOS DE SISTEMAS DE BASES DE DATOS RELACIONALES. CONCEPTOS BÁSICOS

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.

Para llevar a cabo la definición de datos de las bases de datos, el SGBD proveerá de un **Lenguaje de Definición de Datos (DDL: Data Definition Language)** que permitirá definir las estructuras de la base de datos (tablas, vistas e índices).

Para manipular la información existente en las bases de datos, el SGBD proveerá de un **Lenguaje de Manipulación de datos (DML: Data Manipulation Lenguage)**, que permitirá realizar operaciones de consulta y actualización sobre la información de la base de datos.

Cada SGBD tiene sus propios DDL y DML. Veamos más ampliamente los lenguajes de definición de datos y de manipulación de datos.

3. LENGUAJE DE DEFINICIÓN DE DATOS (DDL). TIPOS. CARACTERÍSTICAS.

Un **lenguaje de definición de datos** es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios de la misma llevar a cabo las tareas de definición de las estructuras que almacenarán los datos, así como de los procedimientos o funciones que permitan consultarlos. La definición de la estructura de la base de datos incluye tanto la creación inicial de los diferentes objetos que formarán la base de datos, como el mantenimiento de esa estructura.

Preparador Informática

Habitualmente, los DDL suelen ser lenguajes muy simples con una gramática muy sencilla de manera que permiten describir los datos con facilidad y precisión y sin necesidad de apoyarse en ningún lenguaje de programación.

Gracias a la utilización del DDL obtendremos un conjunto de tablas que se almacenarán en un archivo llamado **diccionario de datos**. En este diccionario de datos contiene "metadatos", es decir, datos sobre nuestra propia base de datos. Cada vez que se añada, modifique o borre información sobre la estructura de la Base de Datos, el diccionario se actualizará para reflejar esos cambios.

3.1. TIPOS

Fundamentalmente existen dos tipos de lenguajes de definición de datos:

- **Interactivo**: Los lenguajes interactivos permiten tratar directamente con el sistema gestor de base de datos por medio de una consola.
- **Incrustado o embebido**: Estos lenguajes se utilizan dentro de un lenguaje anfitrión cuando éste requiera hacer operación contra el sistema gestor de base de datos.

En algunos lenguajes ambos tipos pueden ser solapados, por ejemplo, el SQL se puede usar como lenguaje de consulta interactivo o como lenguaje incrustado.

3.2. CARACTERÍSTICAS

Las características fundamentales de los DDL son:

- **Dependencia del SGBD:** El DDL depende tanto del modelo de base de datos como del SGBD concreto utilizado. En el DDL es donde radican las mayores diferencias entre los distintos SGBD relacionales.
- **Simplicidad:** son lenguajes sencillos, compuestos de pocas instrucciones aplicables a muchos objetos diferentes. Habitualmente se utilizan los mismos verbos para todos los objetos diferentes.
- Son **lenguajes de alto nivel**.
- **Distintos niveles de abstracción**: permiten la definición de datos a varios niveles: externo, lógico e interno.

Preparador Informática

4. LENGUAJE DE MANIPULACIÓN DE DATOS (DML). TIPOS. CARACTERÍSTICAS.

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.

4.1. 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.

Preparador Informática

- **Propio**: Pueden implementar el lenguaje completamente o bien unas pocas operaciones.

4.2. CARACTERÍSTICAS.

Las características más importantes de los DML son:

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

5. LENGUAJE SQL.

5.1. INTRODUCCIÓN.

SQL (Structured Query Language) es el lenguaje fundamental de los SGBD relacionales. Es uno de los lenguajes más utilizados en informática en todos los tiempos. SQL surgió como una evolución de SEQUEL, lenguaje definido por IBM en 1977. ANSI lo definió como estándar unos años más tarde (1986), y fue adoptado también por ISO al año siguiente. Es un lenguaje declarativo y, por tanto, lo más importante es definir qué se desea hacer, y no cómo hacerlo. De esto último ya se encarga el SGBD. Hablamos por tanto de un lenguaje normalizado que nos permite trabajar con cualquier tipo de lenguaje (ASP o PHP) en combinación con cualquier tipo de base de datos (Access, SQL Server, MySQL, Oracle, etc.). El lenguaje SQL puede ser utilizado tanto de forma interactiva (en una consola del SGBD) como inmerso dentro de un lenguaje anfitrión.

SQL posee dos características muy apreciadas, potencia y versatilidad, que contrastan con su facilidad para el aprendizaje, ya que utiliza un lenguaje bastante natural. Es por esto que las instrucciones son muy parecidas a órdenes humanas. Por esta característica se le considera un Lenguaje de Cuarta Generación. Aunque frecuentemente se oye que SQL es un "lenguaje de consulta", eso no es exactamente cierto ya que contiene muchas otras capacidades además de la de consultar la base de datos, como son la definición de la propia estructura de los datos, su manipulación, y la especificación de conexiones seguras. Por tanto, el lenguaje estructurado de consultas SQL es un lenguaje que permite operar con los datos almacenados en las bases de datos relacionales.

5.2. ELEMENTOS DEL LENGUAJE. NORMAS DE ESCRITURA.

El lenguaje SQL está compuesto por comandos, cláusulas, operadores, funciones y literales. Todos estos elementos se combinan en las instrucciones y se utilizan para crear, actualizar y manipular bases de datos.

- **COMANDOS:** Van a ser las instrucciones que se pueden crear en SQL. Se pueden distinguir en tres grupos:
  - **De definición de datos (DDL)**, que permiten crear y definir nuevas bases de datos, tablas, campos, etc.

Comandos DDL. Lenguaje de Definición de Datos.	
Comando:	**Descripción:**
CREATE	Se utiliza para crear nuevas tablas, campos e índices.
DROP	Se utiliza para eliminar tablas e índices.
ALTER	Se utiliza para modificar tablas.

  - **De manipulación de datos (DML)**, que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos, así como inserciones, modificaciones y eliminación de datos.

Comandos DML. Lenguaje de Manipulación de Datos.	
Comando:	**Descripción:**
SELECT	Se utiliza para consultar filas que satisfagan un criterio determinado.
INSERT	Se utiliza para cargar datos en una única operación.
UPDATE	Se utiliza para modificar valores de campos y filas específicos.
DELETE	Se utiliza para eliminar filas de una tabla.

#! Preparador Informática

  - **De control y seguridad de datos (DCL, Data Control Language),** que administran los derechos y restricciones de los usuarios.

Comandos DCL. Lenguaje de Control de Datos.	
Comando:	**Descripción:**
GRANT	Permite dar permisos a uno o varios usuarios o roles para realizar tareas determinadas.
REVOKE	Permite eliminar permisos que previamente se han concedido con GRANT.

- **CLÁUSULAS:** Llamadas también condiciones o criterios, son palabras especiales que permiten modificar el funcionamiento de un comando.

Cláusulas	
Cláusulas:	**Descripción:**
FROM	Se utiliza para especificar la tabla de la que se van a seleccionar las filas.
WHERE	Se utiliza para especificar las condiciones que deben reunir las filas que se van a seleccionar.
GROUP BY	Se utiliza para separar las filas seleccionadas en grupos específicos.
HAVING	Se utiliza para expresar la condición que debe satisfacer cada grupo.
ORDER BY	Se utiliza para ordenar las filas seleccionadas de acuerdo a un orden específico.

- **OPERADORES:** Permiten crear expresiones complejas. Pueden ser aritméticos (+, -, *, /, ...) o lógicos (< , >, , < >, And, Or, etc.).

Operadores de comparación.	
Operadores:	**Descripción:**
<	Menor que.
>	Mayor que.
< >	Distinto de.
< =	Menor o igual.
> =	Mayor o igual.
=	Igual.
BETWEEN	Se utiliza para especificar un intervalo de valores.
LIKE	Se utiliza para comparar.
IN	Se utiliza para especificar filas de una base de datos.

Operadores lógicos.	
Operadores:	**Descripción:**
AND	Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Devuelve el valor contrario de la expresión.

- **FUNCIONES:** Para conseguir valores complejos. Existen muchas funciones distintas, por ejemplo:

Funciones de agregado.	
Función:	**Descripción:**
AVG	Calcula el promedio de los valores de un campo determinado.
COUNT	Devuelve el número de filas de la selección.
SUM	Devuelve la suma de todos los valores de un campo determinado.
MAX	Devuelve el valor más alto de un campo determinado.
MIN	Devuelve el valor mínimo de un campo determinado.

Preparador Informática

- **LITERALES:** Les podemos llamar también constantes y serán valores concretos, como por ejemplo un número, una fecha, un conjunto de caracteres, etc.

Literales	
Literales:	**Descripción:**
23/03/97	Literal fecha.
María	Literal caracteres.
5	Literal número.

Y, además, tendremos que seguir unas normas sencillas pero primordiales:

- Todas las instrucciones terminan con un signo de punto y coma.
- No se distingue entre mayúsculas y minúsculas.
- Cualquier comando puede ser partido con saltos de línea o espacios para facilitar su lectura y comprensión.
- Los comentarios comienzan por /* y terminan con */ (excepto en algunos SGBD).

5.3. SQL COMO DDL.

La primera fase del trabajo con cualquier base de datos comienza con sentencias DDL, puesto que antes de poder almacenar y recuperar información debemos definir las estructuras donde almacenar la información. Las estructuras básicas con las que trabaja SQL son las tablas. Las instrucciones DDL generan acciones que no se pueden deshacer, por eso es conveniente usarlas con precaución y tener copias de seguridad cuando manipulamos la base de datos.

a) Creación y eliminación de base de datos.

Básicamente, la creación de la base de datos consiste en crear las tablas que la componen. Crear una base de datos implica indicar los archivos y ubicaciones que se van a utilizar además de otras indicaciones técnicas y administrativas. Es obvio que todo esto sólo lo puede realizar si se tiene privilegio de Administrador.

Con el estándar de SQL la instrucción a usar sería ***Create Database***, pero cada SGBD tiene un procedimiento para crear las bases de datos. Crearíamos una base de datos con el nombre que se indique a continuación:

```
CREATE DATABASE NombredemiBasedeDatos;
```

En caso de que quisiéramos eliminar una base de datos usamos:

```
DROP DATABASE NombredemiBasedeDatos;
```

#! Preparador Informática

b) Creación de tablas.

Los objetos básicos con los que trabaja SQL son las tablas, que como ya sabemos es un conjunto de filas y columnas cuya intersección se llama celda. Es ahí donde se almacenarán los elementos de información, los datos que queremos recoger. Antes de crear la tabla es conveniente planificar algunos detalles:

- Qué nombre le vamos a dar a la tabla.
- Qué nombre le vamos a dar a cada una de las columnas.
- Qué tipo y tamaño de datos vamos a almacenar en cada columna.
- Qué restricciones tenemos sobre los datos.
- Alguna otra información adicional que necesitemos.

La sintaxis básica del comando que permite crear una tabla es la siguiente:

```
CREATE TABLE [esquema.] NombredeTabla (
columna1 Tipo_Dato [ModificadordeTipo],
```

```
columna2 Tipo_Dato [ModificadordeTipo],
...
columnaN Tipo_Dato [ModificadordeTipo]  ) ;
```

donde:

- *columna1, columna2, ..., columnaN son los nombres de las columnas que contendrá la tabla.*
- *Tipo_Dato indica el tipo de dato de cada columna. Podremos tener, entre otros, datos de tipo Char, number, integer, date y long.*
- *ModificadordeTipo/Restricciones: Una restricción es una condición que una o varias columnas deben cumplir obligatoriamente. Entre otras podemos destacar NULL (Admite valores nulos), NOT NULL (no permite valores nulos) y UNIQUE (no admite valores repetidos). Al definir la tabla podemos indicar cuál de las columnas constituye la clave principal (PRIMARY KEY) que es el cual nos permite distinguir entre varios registros de datos y cuáles son las claves secundarias, indicando el nombre del atributo y la tabla con el que se relaciona (REFERENCES FOREIGN KEY). Las restricciones CKECK exigen la integridad del dominio mediante la limitación de los valores que puede aceptar una columna.*

Ejemplo:

```
CREATE TABLE ALUMNOS (
      NUMERO_MATRICULA INT NOT NULL PRIMARY KEY,
      NOMBRE VARCHAR(15) NOT NULL,
      FECHA_NACIMIENTO DATE,
      DIRECCION VARCHAR(30),
      LOCALIDAD VARCHAR(15)  );
```

#! Preparador Informática

c) Modificación de tablas.

a) Si queremos cambiar el nombre de una tabla:

```
RENAME TABLE NombreViejo TO NombreNuevo;
```

b) Si queremos añadir columnas a una tabla: las columnas se añadirán al final de la tabla.

```
ALTER TABLE NombreTabla ADD
( ColumnaNueva1 Tipo_Dato [ModificadordeTipo],
ColumnaNueva2 Tipo_Dato [ModificadordeTipo],
...
ColumnaNuevaN Tipo_Dato [ModificadordeTipo]  );
```

c) Si queremos eliminar columnas de una tabla: se eliminará la columna indicada sin poder deshacer esta acción. Además, se eliminan todos los datos que contenga la columna.

```
ALTER TABLE NombreTabla DROP COLUMN (Columna1 [, Columna2, …] );
```

d) Si queremos modificar columnas de una tabla: podemos modificar el tipo de dato y las propiedades de una columna. Todos los cambios son posibles si la tabla no contiene datos.

```
ALTER TABLE NombreTabla MODIFY (Columna1 Tipo_Dato [ModificadordeTipo]
[, columna2 Tipo_Dato [ModificadordeTipo] ...] );
```

e) Si queremos renombrar columnas de una tabla:

```
ALTER TABLE NombreTabla RENAME COLUMN NombreAntiguo TO NombreNuevo;
```

f) Podemos añadir y eliminar las siguientes restricciones de una tabla: CHECK, PRIMARY KEY, NOT NULL, FOREIGN KEY y UNIQUE. Para añadir restricciones usamos la orden:

```
ALTER TABLE nombretabla ADD CONSTRAINT nombrerestricción ...
```

Para eliminar restricciones usamos la orden:

```
ALTER TABLE nombretabla DROP CONSTRAINT nombrerestriccion ...
```

d) Eliminación de tablas.

Podemos eliminar una tabla siempre y cuando contemos con privilegios para ello. En tal caso debemos escribir:

```
DROP TABLE NombreTabla [CASCADE];
```

Preparador Informática

CASCADE elimina las restricciones de integridad referencial que remitan a la clave primaria de la tabla borrada.

e) Creación de índices.

Crear índices ayuda a la localización más rápida de la información contenida en las tablas. No es aconsejable que se utilicen índices en campos de tablas pequeñas o campos que se actualicen con mucha frecuencia.

El diseño de índices es un tema bastante complejo para los Administradores de Bases de Datos, ya que una mala elección ocasiona ineficiencia y tiempos de espera elevados. Un uso excesivo de ellos puede dejar a la Base de Datos colgada simplemente con insertar alguna fila. Para crear un índice utilizaríamos la siguiente sentencia:

```
CREATE INDEX NombreIndice ON NombreTabla (Columna1 [, Columna2 ...]);
```

f) Eliminación de índices.

Para eliminar un índice es suficiente con poner la instrucción:

```
DROP INDEX NombreIndice;
```

g) Creación de vistas.

Las vistas permiten definir subconjuntos de datos formados con una o varias tablas y/o vistas. Tiene la apariencia de una tabla, pero ocupa menos espacio que éstas ya que solo se almacena la definición de la vista y no los datos que la forman. Gracias a las vistas podemos mostrar a los usuarios una visión parcial de los datos, de modo que podamos mostrar a los usuarios sólo aquellos datos que son de su interés.

Cuando creamos una vista lo que hacemos es aplicar una consulta sobre la base de datos. La diferencia es que la vista se queda almacenada y podrá ser reutilizada en el futuro. Para crear una vista usamos:

```
CREATE VIEW NombreVista AS Subconsulta;
```

h) Eliminación de vistas.

Las vistas también podrán ser eliminadas de la base de datos. Para ello escribimos lo siguiente:

```
DROP VIEW NombreVista;
```

Preparador Informática

5.4. 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. 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 definición y manipulación de datos. Para ello se han descrito los lenguajes de definición de datos (DDL) y de manipulación de datos (DML) y se han visto los diferentes tipos que existen atendiendo a diferentes clasificaciones. En la segunda mitad del tema se ha introducido el lenguaje SQL, ya que las bases de datos relacionales utilizan como DDL y DML a SQL.

7. 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