TRABAJO DE INVESTIGACION: SQL

PROFESOR: JHON SUAREZ

CURSO: MODELAMIENTO DE LA INFORMACION

ALUMNOS: JAIME LEANDRO LA ROSA
LUCYANA ROMERO CRUCES

AULA – TURNO: 506 – MAÑANA

CICLO: II

NOVIEMBRE 2006

¿Qué es el SQL?

El SQL (Structured query language), lenguaje de consulta estructurado, es un lenguaje surgido de un proyecto de investigación de IBM para el acceso a bases de datos relacionales. Actualmente se ha convertido en un estándar de lenguaje de bases de datos, y la mayoría de los sistemas de bases de datos lo soportan, desde sistemas para ordenadores personales, hasta grandes ordenadores.

Es un lenguaje de base de datos normalizado, utilizado por los diferentes motores de bases de datos para realizar determinadas operaciones sobre los datos o sobre la estructura de los mismos. Pero como sucede con cualquier sistema de normalización hay excepciones para casi todo; de hecho, cada motor de bases de datos tiene sus peculiaridades y lo hace diferente de otro motor, por lo tanto, el lenguaje SQL normalizado (ANSI) no nos servirá para resolver todos los problemas, aunque si se puede asegurar que cualquier sentencia escrita en ANSI será interpretable por cualquier motor de datos.

Como su nombre indica, el SQL nos **permite** realizar **consultas a la base de datos**. Pero el nombre se queda corto ya que SQL además realiza funciones de **definición y manipulación de la base de datos.** Las sentencias SQL se clasifican según su finalidad en dos 'lenguajes' o mejor dicho sublenguajes:

- **DDL** (Data Definition Language), **lenguaje de definición** de datos, incluye órdenes para definir, modificar o borrar las tablas en las que se almacenan los datos y de las relaciones entre estas. (Es el que más varia de un sistema a otro)

- **DML** (Data Manipulation Language), **lenguaje de manipulación** de datos, nos permite recuperar los datos almacenados en la base de datos y también incluye órdenes para permitir al usuario actualizar la base de datos añadiendo nuevos datos, suprimiendo datos antiguos o modificando datos previamente almacenados.

Características del lenguaje

Una sentencia SQL es como una **frase** (escrita en **inglés**) con la que decimos **lo que queremos obtener y de donde obtenerlo**.

Todas las sentencias empiezan con un **verbo** (palabra reservada que indica la acción a realizar), seguido del resto de **cláusulas**, algunas **obligatorias** y otras **opcionales** que completan la frase. Todas las sentencias siguen una **sintaxis** para que se puedan ejecutar correctamente, para describir esa sintaxis utilizaremos un **diagrama sintáctico** como el que se muestra a continuación.

DDL (Data Definition Language), lenguaje de definición de datos

Un **Lenguaje de Definición de Datos** (DDL, por sus siglas en inglés) 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.

El lenguaje de programación SQL, el más difundido entre los gestores de bases de datos, admite las siguientes sentencias de definición: CREATE, DROP y ALTER, cada una de las cuales se puede aplicar a las *tablas*, *vistas*, *procedimientos almacenados* y *triggers* de la Base de Datos.

Otras que se incluyen dentro del DDL, pero que su existencia depende de la implementación del estándar SQL que lleve a cabo el gestor de BD son GRANT y REVOKE, los cuales nos sirven para otorgar permisos o quitarlos, ya sea a usuarios específicos o a un rol creado dentro de la BD.

DML (Data Manipulation Language), lenguaje de manipulación de datos

Una vez creados los esquemas de la base de datos, los usuarios necesitan un lenguaje que les permita manipular los datos de la base de datos: realizar consultas, inserciones, eliminaciones y modificaciones. Este lenguaje es el que se denomina *lenguaje de manejo de datos* (DML).

Hay dos tipos de DML: **los procedurales** y los **no procedurales**. Con un *DML procedural* el usuario (normalmente será un programador) especifica qué datos se necesitan y cómo hay que obtenerlos. Esto quiere decir que el usuario debe especificar todas las operaciones de acceso a datos llamando a los procedimientos necesarios para obtener la información requerida. Estos lenguajes acceden a un registro, lo procesan y basándose en los resultados obtenidos, acceden a otro registro, que también deben procesar. Así se va accediendo a registros y se van procesando hasta que se obtienen los datos deseados. Las sentencias de un DML procedural deben estar embebidas en un lenguaje de alto nivel, ya que se necesitan sus estructuras (bucles, condicionales, etc.) para obtener y procesar cada registro individual. A este lenguaje se le denomina *lenguaje anfitrión*. Las bases de datos jerárquicas y de red utilizan DML procedurales.

Un *DML no procedural* se puede utilizar de manera independiente para especificar operaciones complejas sobre la base de datos de forma concisa. En muchos SGBD se pueden introducir interactivamente instrucciones del DML desde un terminal o bien embeberlas en un lenguaje de programación de alto nivel. Los DML no procedurales permiten especificar los datos a obtener en una consulta o los datos que se deben actualizar, mediante una sola y sencilla sentencia. El usuario o programador especifica qué datos quiere obtener sin decir cómo se debe acceder a ellos. El SGBD traduce las sentencias del DML en uno o varios procedimientos que manipulan los conjuntos de registros necesarios. Esto libera al usuario de tener que conocer cuál es la estructura física de los datos y qué algoritmos se deben utilizar para acceder a ellos. A los DML no procedurales también se les denomina *declarativos*. Las bases de datos relacionales utilizan DML no procedurales, como SQL (Structured Query Language). Los lenguajes no procedurales son más fáciles de aprender y de usar que los procedurales, y el usuario debe realizar menos trabajo, siendo el SGBD quien hace la mayor parte.

La parte de los DML no procedurales que realiza la obtención de datos es lo que se denomina un *lenguaje de consultas*. En general, las órdenes tanto de obtención como de actualización de datos de un DML no procedural se pueden utilizar interactivamente, por lo que al conjunto completo de sentencias del DML se le denomina lenguaje de consultas, aunque es técnicamente incorrecto.

Componentes del SQL

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

Comandos
Existen dos tipos de comandos SQL:
· **Los DDL** que permiten crear y definir nuevas bases de datos, campos e índices.
· **Los DML** que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.

Comandos DDL:
Comando Descripción
CREATE Utilizado para crear nuevas tablas, campos e índices
DROP Empleado para eliminar tablas e índices
ALTER Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

Comandos DML:

Comando Descripción

SELECT Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado
INSERT Utilizado para cargar lotes de datos en la base de datos en una única operación.
UPDATE Utilizado para modificar los valores de los campos y registros especificados
DELETE Utilizado para eliminar registros de una tabla de una base de datos

Cláusulas

Las cláusulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular.

Cláusula Descripción

FROM Utilizada para especificar la tabla de la cual se van a seleccionar los registros
WHERE Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar
GROUP BY Utilizada para separar los registros seleccionados en grupos específicos
HAVING Utilizada para expresar la condición que debe satisfacer cada grupo
ORDER BY Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico

Operadores lógicos

Operador Uso

AND Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdar si alguna de las dos es cierta.
NOT Negación lógica. Devuelve el valor contrario de la expresión.

Operadores de Comparación

Operador Uso

< Menor que
> Mayor que
<> Distinto de
<= Menor ó Igual que
>= Mayor ó Igual que
= Igual que

BETWEEN Utilizado para especificar un intervalo de valores.
LIKE Utilizado en la comparación de un modelo
In Utilizado para especificar registros de una base de datos

Funciones de Agregado

Las funciones de agregado se usan dentro de una cláusula SELECT en grupos de registros para devolver un único valor que se aplica a un grupo de registros.

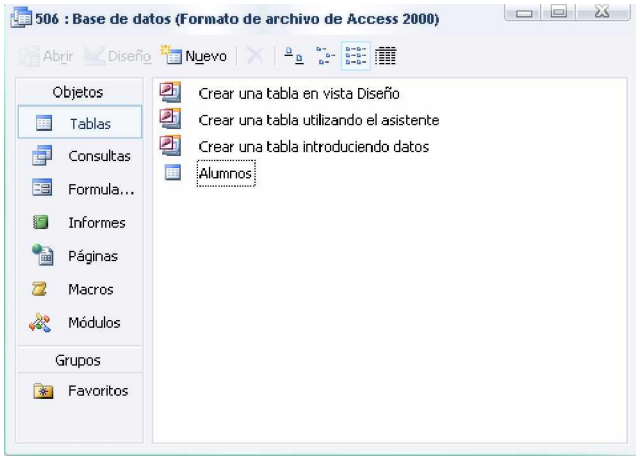
Función Descripción

AVG Utilizada para calcular el promedio de los valores de un campo determinado
COUNT Utilizada para devolver el número de registros de la selección
SUM Utilizada para devolver la suma de todos los valores de un campo determinado
MAX Utilizada para devolver el valor más alto de un campo especificado
MIN Utilizada para devolver el valor más bajo de un campo especificado

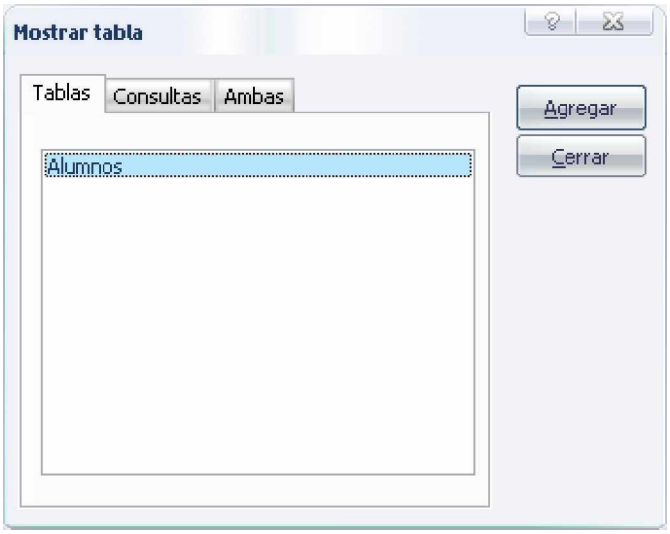
Aplicar comandos SQL utilizando Access 2003

Para esta parte usaremos el motor de base de datos SQL que utiliza el Access 2003, el Microsoft Jet. Para los ejemplos utilizaremos una base de datos la cual contiene una tabla llamada **Alumnos**.

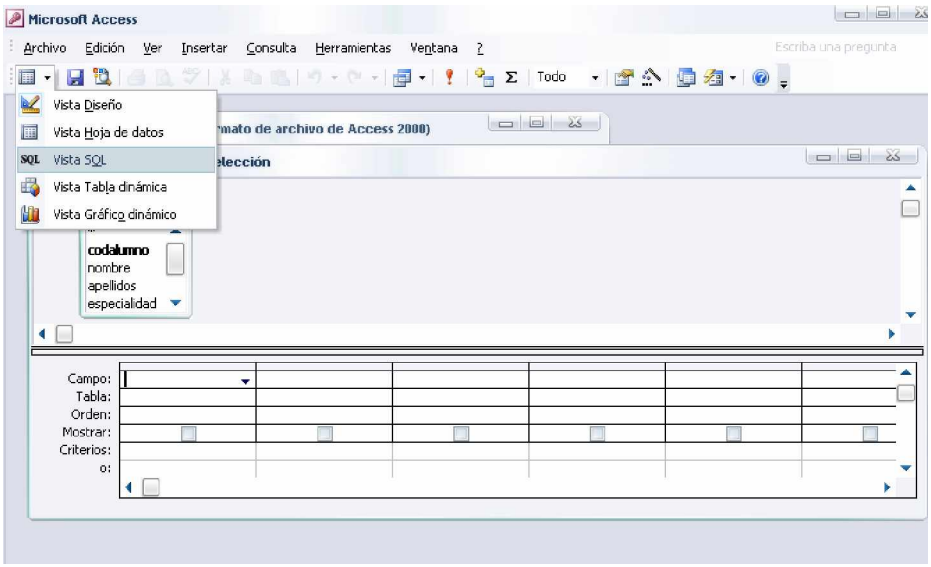
1. Abrir la base de datos **506.mdb** este es el archivo que usaremos para las consultas.
2. A continuación aparece un cuadro de dialogo mostrándonos la tabla que tenemos en la base de datos (Alumnos).



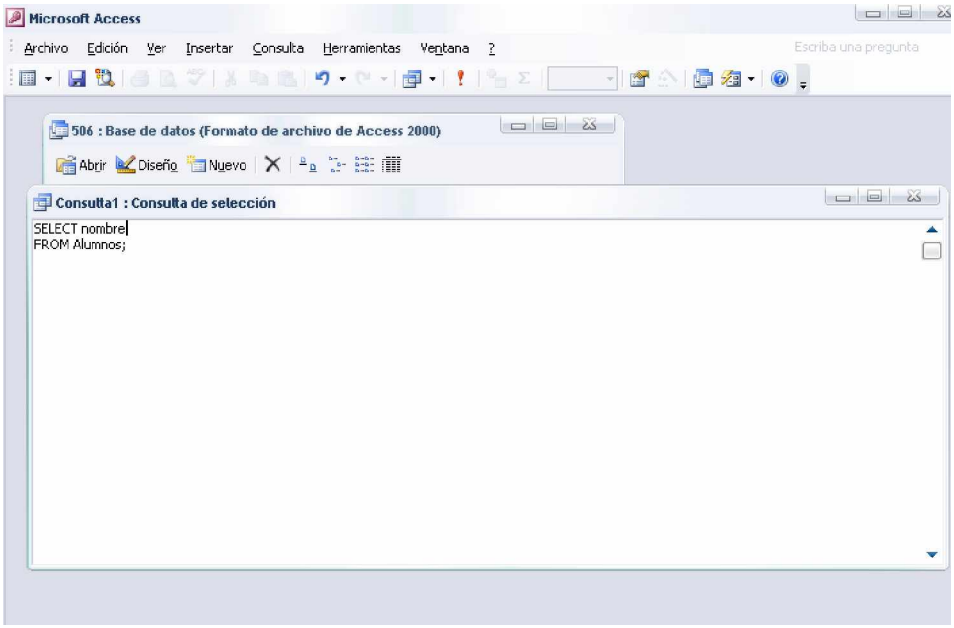
3. En el panel del lado izquierdo en **Objetos** elegimos la opción **Consultas** y al lado derecho elegimos **Crear una consulta en Vista Diseño**, le damos doble clic.
4. En el cuadro de dialogo **Mostrar Tabla** si queremos hacer una consulta escogemos la tabla o tablas en cambio si queremos crear una tabla le damos en cerrar.



5. Luego en la barra estándar en el botón **Vista** escogemos la opción **Vista SQL**.



6. Y por ultimo en la ventana de consultas introducimos los comandos que necesitemos y le damos clic en el botón **Ejecutar**



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

Aplicación de los Comandos Básicos DML

Comandos DML	
Comando	**Descripción**
SELECT	Utilizado para consultar registros de la base de datos que satisfagan un criterio determinado
INSERT	Utilizado para cargar lotes de datos en la base de datos en una única operación.
UPDATE	Utilizado para modificar los valores de los campos y registros especificados
DELETE	Utilizado para eliminar registros de una tabla de una base de datos

En esta parte veremos como se utilizan los comandos básicos y sencillos del **DML**

SELECT

Las consultas de selección se utilizan para indicar al motor de datos que devuelva información de las bases de datos, esta información es devuelta en forma de conjunto de registros que se pueden almacenar en un objeto recordset. Este conjunto de registros es modificable.

La sintaxis básica de una consulta de selección es la siguiente:
SELECT Campos FROM Tabla;
En donde campos es la lista de campos que se deseen recuperar y tabla es el origen de los mismos, por ejemplo:
SELECT nombre, especialidad FROM Alumnos;
Esta consulta devuelve un recordset con el campo Nombrecliente y Telefono de la tabla Cliente.

DELETE

Crea una consulta de eliminación que elimina los registros de una o más de las tablas listadas en la cláusula FROM que satisfagan la cláusula WHERE. Esta consulta elimina los registros completos, no es posible eliminar el contenido de algún campo en concreto. Su sintaxis es: DELETE Tabla.* FROM Tabla WHERE criterio

DELETE es especialmente útil cuando se desea eliminar varios registros. En una instrucción DELETE con múltiples tablas, debe incluir el nombre de tabla (Tabla.*). Si especifica más de una tabla desde la que eliminar registros, todas deben ser tablas de muchos a uno. Si desea eliminar todos los registros de una tabla, eliminar la propia tabla es más eficiente que ejecutar una consulta de borrado.

Se puede utilizar DELETE para eliminar registros de una única tabla o desde varios lados de una relación uno a muchos. Las operaciones de eliminación en cascada en una consulta únicamente eliminan desde varios lados de una relación. Por ejemplo, en la relación entre las tablas Clientes y Pedidos, la tabla Pedidos es la parte de muchos por lo que las operaciones en cascada solo afectaran a la tabla Pedidos. Una consulta de borrado elimina los registros completos, no únicamente los datos en campos específicos. Si desea eliminar valores en un campo especificado, crear una consulta de actualización que cambie los valores a Null. Una vez que se han eliminado los registros utilizando una consulta de borrado, no puede deshacer la operación. Si desea saber qué registros se eliminarán, primero examine los resultados de una consulta de selección que utilice el mismo criterio y después ejecute la consulta de borrado. Mantenga copias de seguridad de sus datos en todo momento. Si elimina los registros equivocados podrá recuperarlos desde las copias de seguridad.
DELETE * FROM Alumnos WHERE especialidad = ‘Computacion’;

INSERT INTO

Agrega un registro en una tabla. Se la conoce como una consulta de datos añadidos. Esta consulta puede ser de dos tipos: Insertar un único registro ó Insertar en una tabla los registros contenidos en otra tabla.

Para insertar un único Registro:

En este caso la sintaxis es la siguiente:

INSERT INTO Tabla (campo1, campo2, .., campoN)
VALUES (valor1, valor2, ..., valorN)

Esta consulta graba en el campo1 el valor1, en el campo2 y valor2 y así sucesivamente. Hay que prestar especial atención a acotar entre comillas simples (') los valores literales (cadenas de caracteres) y las fechas indicarlas en formato mm-dd-aa y entre caracteres de almohadillas (#).

INSERT INTO Alumnos (codalumno, nombre, apellidos, especialidad, edad, curso)
VALUES (1010, 'Jaime', 'Leandro', ‘Computacion’, ‘21’, ‘Visual .Net II’);

UPDATE

Crea una consulta de actualización que cambia los valores de los campos de una tabla especificada basándose en un criterio específico. Su sintaxis es:

UPDATE Tabla SET Campo1=Valor1, Campo2=Valor2,... CampoN=ValorN
WHERE Criterio;

UPDATE es especialmente útil cuando se desea cambiar un gran número de registros o cuando éstos se encuentran en múltiples tablas. Puede cambiar varios campos a la vez.

UPDATE Alumnos SET edad = 20 WHERE nombre = 'Daniel';

Aplicación de los Comandos Básicos DDL

Comandos DDL	
Comando	**Descripción**
CREATE	Utilizado para crear nuevas tablas, campos e índices
DROP	Empleado para eliminar tablas e índices
ALTER	Utilizado para modificar las tablas agregando campos o cambiando la definición de los campos.

En esta parte veremos como se utilizan los comandos básicos y sencillos del **DDL**

CREATE TABLE

Creación de Tablas Nuevas
Si se está utilizando el motor de datos de Microsoft para acceder a bases de datos Access, sólo se puede emplear esta instrucción para crear bases de datos propias de Access. Su sintaxis es:
CREATE TABLE tabla (campo1 tipo (tamaño) índice1, campo2 tipo (tamaño) índice2 , ..., índice multicampo , ...)
En donde:

Parte	Descripción
tabla	Es el nombre de la tabla que se va a crear.
campo1 campo2	Es el nombre del campo o de los campos que se van a crear en la nueva tabla. La nueva tabla debe contener, al menos, un campo.
tipo	Es el tipo de datos de campo en la nueva tabla.
tamaño	Es el tamaño del campo sólo se aplica para campos de tipo texto.
índice1 índice2	Es una cláusula CONSTRAINT que define el tipo de índice a crear. Esta cláusula en opcional.
índice multicampos	Es una cláusula CONSTRAINT que define el tipo de índice multicampos a crear. Un índice multicampo es aquel que está indexado por el contenido de varios campos. Esta cláusula en opcional.

CREATE TABLE Profesor (Codprofesor INTEGER, nombre TEXT (15), apellidos TEXT (25));
Crea una nueva tabla llamada Profesor con dos campos, uno llamado Nombre de tipo texto y longitud 15 y otro llamado apellidos con longitud 25.

ALTER TABLE

La sentencia ALTER TABLE sirve para modificar la estructura de una tabla que ya existe. Mediante esta instrucción podemos añadir columnas nuevas, eliminar columnas. Ten cuenta que cuando eliminamos una columna se pierden todos los datos almacenados en ella.
También nos permite crear nuevas restricciones o borrar algunas existentes. La sintaxis puede parecer algo complicada pero sabiendo el significado de las palabras reservadas la sentencia se aclara bastante; ADD (añade), ALTER (modifica), DROP (elimina), COLUMN (columna), CONSTRAINT (restricción).

La sintaxis es la siguiente:
ALTER TABLE *tabla* ADD COLUMN *Campo tipo* (*tamaño*)

ALTER TABLE Profesor ADD COLUMN edad INTEGER;

DROP TABLE

La sentencia DROP TABLE sirve para eliminar una tabla. No se puede eliminar una tabla si está abierta, tampoco la podemos eliminar si el borrado infringe las reglas de integridad referencial (si interviene como tabla padre en una relación y tiene registros relacionados).

La sintaxis es la siguiente:

DROP TABLE tabla

Ejemplo:

DROP TABLE Profesor

Elimina de la base de datos la tabla Profesor.

Datos Complementarios

Cláusulas

Las cláusulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular.

Cláusula	Descripción
FROM	Utilizada para especificar la tabla de la cual se van a seleccionar los registros
WHERE	Utilizada para especificar las condiciones que deben reunir los registros que se van a seleccionar
GROUP BY	Utilizada para separar los registros seleccionados en grupos específicos
HAVING	Utilizada para expresar la condición que debe satisfacer cada grupo
ORDER BY	Utilizada para ordenar los registros seleccionados de acuerdo con un orden específico

Operadores Lógicos

Operador	Uso
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.

Operadores de Comparación

Operador	Uso
<	Menor que
>	Mayor que
<>	Distinto de
<=	Menor ó Igual que
>=	Mayor ó Igual que
=	Igual que
BETWEEN	Utilizado para especificar un intervalo de valores.
LIKE	Utilizado en la comparación de un modelo
In	Utilizado para especificar registros de una base de datos

Funciones de Agregado

Las funciones de agregado se usan dentro de una cláusula SELECT en grupos de registros para devolver un único valor que se aplica a un grupo de registros.

Función	Descripción
AVG	Utilizada para calcular el promedio de los valores de un campo determinado
COUNT	Utilizada para devolver el número de registros de la selección
SUM	Utilizada para devolver la suma de todos los valores de un campo determinado
MAX	Utilizada para devolver el valor más alto de un campo especificado
MIN	Utilizada para devolver el valor más bajo de un campo especificado

Tipos de Datos

Los tipos de datos SQL se clasifican en 13 tipos de datos primarios y de varios sinónimos Válidos reconocidos por dichos tipos de datos.
Tipos de datos primarios:

Tipo de Datos	Longitud	Descripción
BINARY	1 byte	Para consultas sobre tabla adjunta de productos de bases de datos que definen un tipo de datos Binario.
BIT	1 byte	Valores Si/No ó True/False
BYTE	1 byte	Un valor entero entre 0 y 255.
COUNTER	4 bytes	Un número incrementado automáticamente (de tipo Long)
CURRENCY	8 bytes	Un entero escalable entre 922.337.203.685.477,5808 y 922.337.203.685.477,5807.
DATETIME	8 bytes	Un valor de fecha u hora entre los años 100 y 9999.
SINGLE	4 bytes	Un valor en punto flotante de precisión simple con un rango de $-3.402823*10^{38}$ a $-1.401298*10^{-45}$ para valores negativos, $1.401298*10^{-45}$ a $3.402823*10^{38}$ para valores positivos, y 0.
DOUBLE	8 bytes	Un valor en punto flotante de doble precisión con un rango de $-1.79769313486232*10^{308}$ a $-4.94065645841247*10^{-324}$ para valores negativos, $4.94065645841247*10^{-324}$ a $1.79769313486232*10^{308}$ para valores positivos, y 0.
SHORT	2 bytes	Un entero corto entre -32,768 y 32,767.
LONG	4 bytes	Un entero largo entre -2,147,483,648 y 2,147,483,647.
LONGTEXT	1 byte por carácter	De cero a un máximo de 1.2 gigabytes.
LONGBINARY	Según se necesite	De cero 1 gigabyte. Utilizado para objetos OLE.
TEXT	1 byte por caracter	De cero a 255 caracteres.