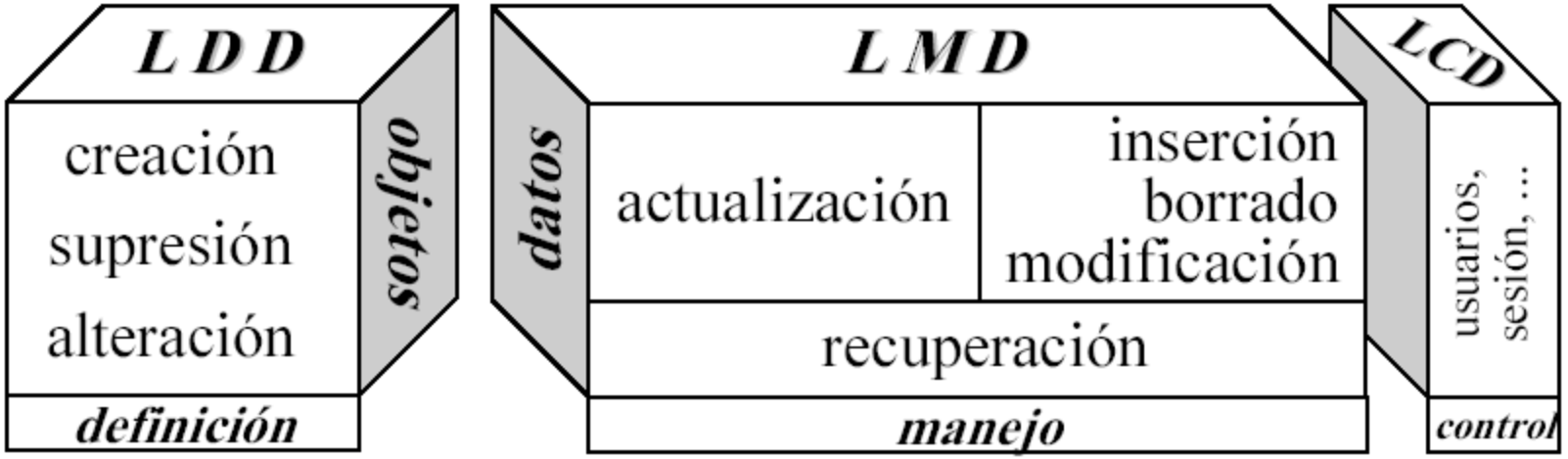
Clase lab3:
Introducción Lenguaje SQL
Continuación....

Prof. Ania Cravero

Lenguaje SQL (Structured Query Language)

- Instrucciones del lenguaje SQL: divididas en dos tipos
  - Lenguaje de Definición de Datos LDD o DDL
  - Lenguaje de Manipulación de Datos LMD o MDL
- Ambos conjuntos son complementarios



Lenguaje SQL: LDD o DDL

Comandos DLL

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.

Lenguaje SQL: LMD o MDL

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

Lenguaje SQL: MDL

- 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

Lenguaje SQL: MDL

- Operadores Lógicos y de Comparación

Operador	Uso
AND	Es el "y" lógico. Evalua 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.

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

Lenguaje SQL: MDL

- **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

Consultas de Selección

- 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.
- Sintaxis básica
  - **SELECT campos FROM Tablas WHERE condición**

- Ejemplo
  - **SELECT Nombre, Telefono FROM Clientes**;

Consultas de Selección

- Ordenar los registros: Adicionalmente se puede especificar el orden en que se desean recuperar los registros de las tablas mediante la claúsula ORDER BY Lista de Campos. En donde Lista de campos representa los campos a ordenar.
- Ejemplo:
  - **SELECT CodigoPostal, Nombre, Telefono FROM Clientes ORDER BY Nombre;**
- Se pueden ordenar los registros por mas de un campo, como por ejemplo:
  - **SELECT CodigoPostal, Nombre, Telefono FROM Clientes ORDER BY CodigoPostal, Nombre;**
- Incluso se puede especificar el orden de los registros: ascendente mediante la claúsula (ASC -se toma este valor por defecto) ó descendente (DESC)
  - **SELECT CodigoPostal, Nombre, Telefono FROM Clientes ORDER BY CodigoPostal DESC , Nombre ASC;**

Consultas de Selección

- Consultas con predicado: El predicado se incluye entre la clausula y el primer nombre del campo a recuperar, los posibles predicados son:

Predicado	Descripción
ALL	Devuelve todos los campos de la tabla
TOP	Devuelve un determinado número de registros de la tabla
DISTINCT	Omite los registros cuyos campos seleccionados coincidan totalmente
DISTINCTROW	Omite los registros duplicados basandose en la totalidad del registro y no sólo en los campos seleccionados.

Ejemplos:
SELECT ALL FROM Empleados;
SELECT * FROM Empleados;

Ejemplo:
SELECT TOP 25 Nombre, Apellido
FROM Estudiantes
ORDER BY Nota DESC;

Consultas de Selección

- DISTINCT: Omite los registros que contienen datos duplicados en los campos seleccionados. Para que los valores de cada campo listado en la instrucción SELECT se incluyan en la consulta deben ser únicos.
- Por ejemplo, varios empleados listados en la tabla Empleados pueden tener el mismo apellido. Si dos registros contienen López en el campo Apellido, la siguiente instrucción SQL devuelve un único registro:
  - SELECT DISTINCT Apellido FROM Empleados;

- DISTINCTROW: Devuelve los registros diferentes de una tabla; a diferencia del predicado anterior que sólo se fijaba en el contenido de los campos seleccionados, éste lo hace en el contenido del registro completo independientemente de los campo indicados en la cláusula SELECT.
  - SELECT DISTINCTROW Apellido FROM Empleados;

Consultas de Selección

- ALIAS: En determinadas circunstancias es necesario asignar un nombre a alguna columna determinada de un conjunto devuelto, otras veces por simple capricho o por otras circunstancias. Para resolver todas ellas tenemos la palabra reservada AS que se encarga de asignar el nombre que deseamos a la columna deseada. Tomado como referencia el ejemplo anterior podemos hacer que la columna devuelta por la consulta, en lugar de llamarse apellido (igual que el campo devuelto) se llame Empleado.
- En este caso procederíamos de la siguiente forma:
  - SELECT DISTINCTROW Apellido AS Empleado FROM Empleados;

Consultas de Selección

- Criterios de Selección: se estudiarán las posibilidades de filtrar los registros con el fin de recuperar solamente aquellos que cumplan una condiciones preestablecidas.
- Operadores Lógicos: Los operadores lógicos soportados por SQL son: AND, OR, XOR, Eqv, Imp, Is y Not. A excepción de los dos últimos todos poseen la siguiente sintaxis:
  - <expresión1> operador <expresión2>
- El último operador denominado Is se emplea para comparar dos variables de tipo objeto <Objeto1> Is <Objeto2>. este operador devuelve verdad si los dos objetos son iguales
- Ejemplos:
  - SELECT * FROM Empleados WHERE Edad > 25 AND Edad < 50;
  - SELECT * FROM Empleados WHERE (Edad > 25 AND Edad < 50) OR Sueldo =100;
  - SELECT * FROM Empleados WHERE NOT Estado = 'Soltero';
  - SELECT * FROM Empleados WHERE (Sueldo > 100 AND Sueldo < 500) OR (Provincia = 'Madrid' AND Estado = 'Casado');

Consultas de Selección

- Intervalos de valores: Para indicar que deseamos recuperar los registros según el intervalo de valores de un campo emplearemos el operador Between cuya sintaxis es:
  - **campo [Not] Between valor1 And valor2 (la condición Not es opcional)**
- En este caso la consulta devolvería los registros que contengan en "campo" un valor incluido en el intervalo valor1, valor2 (ambos inclusive). Si anteponemos la condición Not devolverá aquellos valores no incluidos en el intervalo.
- **Ejemplos**:
  - SELECT * FROM Pedidos WHERE CodPostal Between 28000 And 28999; (Devuelve los pedidos realizados en la provincia de Madrid)
  - SELECT IIf(CodPostal Between 28000 And 28999, 'Provincial', 'Nacional') FROM Editores; (Devuelve el valor 'Provincial' si el código postal se encuentra en el intervalo, 'Nacional' en caso contrario)

Consultas de Selección

- El Operador Like: Se utiliza para comparar una expresión de cadena con un modelo en una expresión SQL. Su sintaxis es:
  - expresión Like modelo
- En donde expresión es una cadena modelo o campo contra el que se compara expresión. Se puede utilizar el operador Like para encontrar valores en los campos que coincidan con el modelo especificado. Por modelo puede especificar un valor completo (Ana María), o se pueden utilizar caracteres comodín como los reconocidos por el sistema operativo para encontrar un rango de valores (Like An*).
- El ejemplo siguiente devuelve los datos que comienzan con la letra P seguido de cualquier letra entre A y F y de tres dígitos:
  - **Like 'P[A-F]###'**
- Este ejemplo devuelve los campos cuyo contenido empiece con una letra de la A a la D seguidas de cualquier cadena.
  - **Like '[A-D]*'**

Expresiones con diferentes modelos para el operador LIKE

Tipo de coincidencia	Modelo Planteado	Coincide	No coincide
Varios caracteres	'a*a'	'aa', 'aBa', 'aBBBa'	'aBC'
Carácter especial	'a[*]a'	'a*a'	'aaa'
Varios caracteres	'ab*'	'abcdefg', 'abc'	'cab', 'aab'
Un solo carácter	'a?a'	'aaa', 'a3a', 'aBa'	'aBBBa'
Un solo dígito	'a#a'	'a0a', 'a1a', 'a2a'	'aaa', 'a10a'
Rango de caracteres	'[a-z]'	'f', 'p', 'j'	'2', '&'
Fuera de un rango	'[!a-z]'	'9', '&', '%'	'b', 'a'
Distinto de un dígito	'[!0-9]'	'A', 'a', '&', '~'	'0', '1', '9'
Combinada	'a[!b-m]#'	'An9', 'az0', 'a99'	'abc', 'aj0'

Consultas de Selección

- El Operador In: Este operador devuelve aquellos registros cuyo campo indicado coincide con alguno de los en una lista. Su sintaxis es:
  - expresión [Not] In(valor1, valor2, . . .)
- Ejemplo:
  - **SELECT * FROM Pedidos WHERE Provincia In ('Madrid', 'Barcelona', 'Sevilla');**

- La cláusula WHERE: La cláusula WHERE puede usarse para determinar qué registros de las tablas enumeradas en la cláusula FROM aparecerán en los resultados de la instrucción SELECT. Si no se emplea esta cláusula, la consulta devolverá todas las filas de la tabla. WHERE es opcional, pero cuando aparece debe ir a continuación de FROM.
- Ejemplo:
  - **SELECT Apellidos, Salario FROM Empleados WHERE Salario > 21000;**
  - **SELECT Id_Producto, Existencias FROM Productos WHERE Existencias <= Nuevo_Pedido;**

Agrupamiento de Registros

- GROUP BY: Combina los registros con valores idénticos, en la lista de campos especificados, en un único registro. Para cada registro se crea un valor sumario si se incluye una función SQL agregada, como por ejemplo Sum o Count, en la instrucción SELECT. Su sintaxis es:
  - **SELECT campos FROM tabla WHERE criterio GROUP BY campos del grupo**
- Se utiliza la cláusula **WHERE** para excluir aquellas filas que no desea agrupar, y la cláusula **HAVING** para filtrar los registros una vez agrupados.
- Ejemplo:
  - SELECT Id_Familia, Sum(Stock) FROM Productos GROUP BY Id_Familia;
  - SELECT Id_Familia Sum(Stock) FROM Productos GROUP BY Id_Familia HAVING Sum(Stock) > 100 AND NombreProducto Like BOS*;

Agrupamiento de Registros

- AVG: Calcula la media aritmética de un conjunto de valores contenidos en un campo especificado de una consulta. Su sintaxis es la siguiente
  - **Avg(expr)**
- En donde expr representa el campo que contiene los datos **numéricos para los que se desea calcular la media** o una expresión que realiza un cálculo utilizando los datos de dicho campo. La media calculada por Avg es la media aritmética (la suma de los valores dividido por el número de valores). La función Avg no incluye ningún campo Null en el cálculo.
- Ejemplo:
  - SELECT Avg(Gastos) AS Promedio FROM Pedidos WHERE Gastos > 100;

Agrupamiento de Registros

- **Count:** Calcula el número de registros devueltos por una consulta. Su sintaxis es la siguiente:
  - Count(expr)
- En donde expr contiene el nombre del campo que desea contar. Los operandos de expr pueden incluir el nombre de un campo de una tabla, una constante o una función (la cual puede ser intrínseca o definida por el usuario pero no otras de las funciones agregadas de SQL). Puede contar cualquier tipo de datos incluso texto.
- Aunque expr puede realizar un cálculo sobre un campo, Count simplemente cuenta el número de registros sin tener en cuenta qué valores se almacenan en los registros. La función Count no cuenta los registros que tienen campos null a menos que expr sea el carácter comodín asterisco (*). Si utiliza un asterisco, Count calcula el número total de registros, incluyendo aquellos que contienen campos null.
- **Ejemplos**:
  - SELECT Count(*) AS Total FROM Pedidos; (Si expr identifica a múltiples campos, la función Count cuenta un registro sólo si al menos uno de los campos no es Null. Si todos los campos especificados son Null, no se cuenta el registro. Hay que separar los nombres de los campos con ampersand (&)).
  - SELECT Count(FechaEnvío & Transporte) AS Total FROM Pedidos;

Agrupamiento de Registros

- Sum: Devuelve la suma del conjunto de valores contenido en un campo especifico de una consulta. Su sintaxis es:
  - **Sum(expr)**
- En donde expr respresenta el nombre del campo que contiene los datos que desean sumarse o una expresión que realiza un cálculo utilizando los datos de dichos campos. Los operandos de expr pueden incluir el nombre de un campo de una tabla, una constante o una función (la cual puede ser intrínseca o definida por el usuario pero no otras de las funciones agregadas de SQL).
- **Ejemplo**:
  - SELECT Sum(PrecioUnidad * Cantidad) AS Total FROM DetallePedido;

- Otras funciones de agregado:
  - VAR
  - StDev
  - Max, Min

Consultas de Acción: DELETE

- 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
- **Ejemplo**:
  - DELETE * FROM Empleados WHERE Cargo = 'Vendedor';

Consultas de Acción: INSERT

- **INSERT INTO**: Agrega un registro en una tabla. Se la conoce como una consulta de datos añadidos.
- **Para insertar un único Registro**: En este caso la sintaxis es la siguiente:
  - INSERT INTO Tabla (campo1, campo2, .., campoN) VALUES (valor1, valor2, …, valorN)
- **Para insertar Registros de otra Tabla**: En este caso la sintaxis es:
  - INSERT INTO Tabla [IN base_externa] (campo1, campo2, …, campoN) SELECT TablaOrigen.campo1, TablaOrigen.campo2, …, TablaOrigen.campoN FROM TablaOrigen
- Ejemplo:
- INSERT INTO Clientes SELECT Clientes_Viejos.* FROM Clientes_Nuevos;
- INSERT INTO Empleados (Nombre, Apellido, Cargo) VALUES ('Luis', 'Sánchez', 'Becario');
- INSERT INTO Empleados SELECT Vendedores.* FROM Vendedores WHERE Fecha_Contratacion < Now() - 30;

Consultas de Acción: UPDATE

- 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.
- **Ejemplo**: El ejemplo siguiente incrementa los valores Cantidad pedidos en un 10 por ciento y los valores Transporte en un 3 por ciento para aquellos que se hayan enviado al Reino Unido.:
  - UPDATE Pedidos SET Pedido = Pedidos * 1.1, Transporte = Transporte * 1.03 WHERE PaisEnvío = 'ES';

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.

Tipos de datos

- Más tipos de datos primarios

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.