Tema 4:
Dinámica del Modelo Relacional.
El lenguaje SQL

Departamento de

Ciencias de la Computación e Inteligencia Artificial

UNIVERSIDAD DE SEVILLA

Bases de Datos

Curso 2007--08

El Lenguaje SQL

El *Lenguaje de Consulta Estructurado* (SQL = Structured Query Language) es un lenguaje de acceso a datos normalizado utilizado por muchos motores de bases de datos (entre otros, el de Microsoft Jet, que es el que usa *Access*).

1 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.

2 Comandos

Los comandos del SQL pueden dividirse en tres grupos:

- Comandos de definición de datos (DDL = *Data Definition Language*), que permiten crear y definir nuevas bases de datos, campos, …
- Comandos de manipulación de datos (DML = *Data Manipulation Language*), que permiten generar consultas para ordenar, filtrar y extraer datos de la base de datos.
- Comandos de control y seguridad de datos, que gobiernan los privilegios de los usuarios, los controles de acceso, …

Los principales comandos del lenguaje SQL son:

Comandos de DDL	
Comando	**Descripción**
CREATE	Encargado de crear nuevas tablas, campos, …
DROP	Encargado de eliminar tablas
ALTER	Encargado de modificar las tablas, agregando campos o cambiando la definición de los campos

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

3 Condiciones o criterios

Por medio de ciertos modificadores, llamados **cláusulas**, se consigue generar criterios con el fin de definir los datos que se desea seleccionar o manipular.

Cláusula	Descripción
FROM	Sirve para especificar la tabla de la cual se van a seleccionar los registros
WHERE	Sirve para especificar las condiciones que deben reunir los registros que se van a seleccionar
GROUP BY	Sirve para especificar un criterio adicional por el que agrupar los registros seleccionados
HAVING	Sirve para expresar la condición que debe satisfacer cada grupo anterior
ORDER BY	Sirve para ordenar los registros seleccionados de acuerdo con el orden especificado

4 Operadores Lógicos

Operador	Descripción
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.

5 Operadores de Comparación

Operador	Descripción
<	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

6 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 (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 una nueva tabla.

Consultas de selección básicas

La sintaxis más sencilla de una consulta de selección es la siguiente:

SELECT *Campos* FROM *Tabla*;

Donde *campos* es la lista de campos que se deseen recuperar y *tabla* es el origen de los mismos. Por ejemplo, la consulta

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

devuelve una tabla temporal con los campos *nombre* y *teléfono* de la tabla *clientes*.

Ordenar los registros (ORDER BY)

Adicionalmente se puede especificar el orden en que se desean recuperar los registros de las tablas mediante la cláusula ORDER BY *Lista-Campos*. Donde *Lista-campos* representa los campos a ordenar. Por ejemplo, la consulta

SELECT *CodigoPostal*, *Nombre*, *Telefono* FROM *Clientes* ORDER BY *Nombre*;

devuelve los campos *CodigoPostal*, *Nombre* y *Telefono* de la tabla *Clientes* ordenados por el campo *Nombre*.

Se pueden ordenar los registros por más de un campo. Por ejemplo:

SELECT *CodigoPostal*, *Nombre*, *Telefono* FROM *Clientes* ORDER BY *CodigoPostal*, *Nombre*;

Incluso se puede especificar el orden de los registros: *ascendente* mediante la cláusula ASC (valor por defecto) o *descendente* DESC. Por ejemplo:

SELECT *CodigoPostal*, *Nombre*, *Telefono* FROM *Clientes* ORDER BY *CodigoPostal* DESC , *Nombre* ASC;

Consultas con Predicado (ALL, TOP, DISTINCT)

En cuanto al conjunto de registros seleccionados, estos modificadores, que se incluyen entre SELECT y el primer nombre del campo a recuperar, provocan las siguientes acciones:

Predicado	Descripción
ALL	Devuelve todos los campos de la tabla (valor por defecto)
TOP	Devuelve un determinado número de registros de la tabla
DISTINCT	Omite repeticiones de registros cuyos campos seleccionados coincidan totalmente
DISTINCTROW	Omite repeticiones de registros basándose en la totalidad del registro y no sólo en los campos seleccionados.

ALL / *
Es el valor por defecto. El Motor de base de datos selecciona todos los registros que cumplen las condiciones de la instrucción SQL. No es conveniente abusar de este predicado ya que obligamos al motor de la base de datos a analizar la estructura de la tabla para averiguar los campos que contiene, es mucho más rápido indicar el listado de campos deseados. Se suele sustituir por el símbolo * . Por ejemplo:

SELECT ALL FROM *Empleados*;
SELECT * FROM *Empleados*;

TOP
Devuelve un cierto número de registros que corresponden al principio o al final de un rango especificado por una cláusula ORDER BY. Por ejemplo, supongamos que queremos recuperar los nombres de los 25 estudiantes con mejor nota del curso 1994. Para ello, emplearemos la consulta:

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

Si no se incluye la cláusula ORDER BY, la consulta devolverá un conjunto arbitrario de 25 registros de la tabla Estudiantes. Además, el predicado TOP no elige entre valores iguales. En el ejemplo anterior, si la nota número 25 y la 26 son iguales, la consulta devolverá 26 registros en lugar de 25.

Se puede utilizar la palabra reservada PERCENT para devolver un cierto porcentaje de registros que caen al principio o al final de un rango especificado por la cláusula ORDER BY. Supongamos que, en lugar de los 25 primeros estudiantes, deseamos el 10 por ciento del curso:

SELECT TOP 10 PERCENT *Nombre*, *Apellido* FROM *Estudiantes* ORDER BY *Nota* DESC;

El valor que va a continuación de TOP debe ser de tipo Entero sin signo.

DISTINCT

Omite los registros que contienen datos duplicados en los campos seleccionados. 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*;

Con otras palabras, el predicado DISTINCT devuelve aquellos registros cuyos campos indicados en la cláusula SELECT posean un contenido diferente.

DISTINCTROW

Devuelve los registros diferentes de una tabla; y, 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 campos indicados en la cláusula SELECT). Por ejemplo, considérese la consulta:

SELECT DISTINCTROW *Apellido* FROM *Empleados*;

Si la tabla *empleados* contiene dos registros: Antonio López y Marta López, el ejemplo del predicado DISTINCT devuelve un único registro con el valor López en el campo *Apellido* (ya que busca no duplicados en dicho campo). Sin embargo, la consulta con del predicado DISTINCTROW devuelve *dos* registros con el valor López en el *apellido* (ya que se buscan no duplicados en el registro completo).

Alias (AS)

En determinadas circunstancias es necesario asignar un nombre nuevo a alguna columna determinada de un conjunto de registros devuelto por una consulta. Para ello, usaremos la palabra reservada AS, que se encarga de asignar el nombre que deseamos a la columna deseada. En 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*;

Criterios de Selección

Los métodos empleados hasta el momento para de recuperar un conjunto de registros de una tabla devuelven todos los registros de la tabla. A continuación se estudiarán las posibilidades de filtrar los registros con el fin de recuperar solamente aquellos que cumplan ciertas condiciones preestablecidas.
Antes de comenzar debemos tener presente que:

- Cada vez que se desee establecer una condición referida a un campo de texto la condición de búsqueda debe ir encerrada entre comillas simples.
- Las fechas se deben escribir siempre en formato *mm-dd-aa* en donde *mm* representa el mes, *dd* el día y *aa* el año. Además la fecha debe ir encerrada entre almohadillas (#). Por ejemplo si deseamos referirnos al día 3 de Septiembre de 1995 debemos escribir: #09-03-95# ó #9-3-95#.

Operadores Lógicos.

Algunos operadores lógicos soportados por SQL son: AND, OR y NOT. Los dos primeros poseen la sintaxis:

<expresión1> operador *<expresión2>*

En donde *expresión1* y *expresión2* son las condiciones a evaluar, el resultado de la operación varía en función del operador lógico

Si a cualquiera de las anteriores condiciones le anteponemos el operador NOT el resultado de la operación será el contrario al devuelto sin el operador NOT.

Por ejemplo, considérense las consultas:

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');

Intervalos de Valores (BETWEEN)

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.
Por ejemplo:

SELECT * FROM *Pedidos* WHERE *CodPostal* BETWEEN 28000 AND 28999;
(Devuelve los pedidos cuyo código postal esté en el intervalo [28000,28999])

SELECT * FROM *Pedidos* WHERE *CodPostal* NOT BETWEEN 28000 AND 28999;
(Devuelve los pedidos cuyo código postal sea menor que 28000 ó bien mayor que 28999)

El Operador LIKE

Se utiliza para comparar una expresión de cadena con un modelo en una expresión SQL. Su sintaxis es:

expresión LIKE *modelo*

Donde *expresión* es una cadena o campo, y *modelo* un patrón contra el que se compara expresión.
Se puede utilizar el operador LIKE para encontrar valores en los campos que coincidan con el modelo especificado. Como *modelo*, se puede especificar un valor completo (‘Ana María’), o bien se pueden utilizar caracteres comodín como los reconocidos por el sistema operativo para encontrar un rango de valores (por ejemplo, LIKE An*).
Por ejemplo, si se introduce LIKE ‘C*’ en una consulta SQL, la consulta devuelve todos los valores de campo que comiencen por la letra C.
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]*'

El Operador IN

Este operador devuelve los registros cuyo campo indicado coincide con alguno de los dados en una lista. Su sintaxis es:

expresión [NOT] IN (*valor1*, *valor2*, . . .)

Por ejemplo:

SELECT * FROM *Pedidos* WHERE *Provincia* IN ('Madrid', 'Cádiz', 'Sevilla');

El cuantificador existencial (EXISTS)

La expresión siguiente:

WHERE EXISTS (SELECT …. FROM ....)

es verdadera si y sólo si el resultado de evaluar la consulta especificada por SELECT …. FROM .... **no** es el conjunto vacío.

La cláusula WHERE

Como hemos visto, la cláusula WHERE se usa para determinar qué registros de las tablas enumeradas en la cláusula FROM aparecerán en los resultados de la instrucción SELECT. Después de escribir esta cláusula se deben especificar las condiciones expuestas en los apartados anteriores. 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.
A continuación se listan algunos ejemplos:

SELECT *Apellidos*, *Salario* FROM *Empleados* WHERE *Salario* > 21000;

SELECT *Apellidos*, *Nombre* FROM *Empleados* WHERE *Apellidos* = 'King';

SELECT *Apellidos*, *Nombre* FROM *Empleados* WHERE *Apellidos* Like 'S*';

SELECT *Apellidos*, *Salario* FROM *Empleados* WHERE *Salario* Between 200 And 300;

SELECT *Apellidos*, *Salario* FROM *Empl* WHERE *Apellidos* Between 'Lon' And 'Tol';

SELECT *Id_Pedido*, *Fecha_Pedido* FROM *Pedidos* WHERE *Fecha_Pedido* Between #1-1-94# And #30-6-94#;

SELECT *Apellidos*, *Nombre*, *Ciudad* FROM *Empleados* WHERE *Ciudad* IN('Sevilla', 'Los Ángeles', 'Barcelona') OR Nombre NOT LIKE ‘*B*’;

Agrupamiento de Registros y funciones agregadas

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:

SELECTcampos FROM *tabla* WHERE *criterio* GROUP BY *campos del grupo*

GROUP BY es opcional. Los valores de resumen se omiten si no existe una función SQL agregada en la instrucción SELECT. Se utiliza la cláusula WHERE para excluir aquellas filas que no desea agrupar, y la cláusula HAVING para filtrar los registros una vez agrupados.

Un campo de la lista de campos GROUP BY puede referirse a cualquier campo de las tablas que aparecen en la cláusula FROM, incluso si el campo no esta incluido en la instrucción SELECT, siempre y cuando la instrucción SELECT incluya al menos una función SQL agregada.

Todos los campos de la lista de campos de SELECT deben o bien incluirse en la cláusula GROUP BY o como argumentos de una función SQL agregada.

SELECT *Id_Familia*, Sum(*Stock*)FROM *Productos* GROUP BY *Id_Familia*;

Una vez que GROUP BY ha combinado los registros, HAVING muestra cualquier registro agrupado por la cláusula GROUP BY que satisfaga las condiciones de la cláusula HAVING.
HAVING es similar a WHERE, determina qué registros se seleccionan. Una vez que los registros se han agrupado utilizando GROUP BY, HAVING determina cuales de ellos se van a mostrar.

SELECT *Id_Familia*, Sum(*Stock*) FROM *Productos* GROUP BY *Id_Familia* HAVING Sum(*Stock*) > 100 AND *NombreProducto* Like BOS*;

AVG

Calcula la media aritmética de un conjunto de valores contenidos en un campo especificado de una consulta. Su sintaxis es: Avg(*expr*)
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).

SELECT Avg(*Gastos*) AS *Promedio* FROM *Pedidos* WHERE *Gastos* > 100;

Count

Calcula el número de registros devueltos por una consulta. Su sintaxis es la siguiente: Count(*expr*) .

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.

La función Count no cuenta los registros que tienen campos null a menos que *expr* sea el carácter comodín asterisco :

```
SELECT Count(*) AS Total FROM Pedidos;
```

Max, Min

Devuelven el mínimo o el máximo de un conjunto de valores contenidos en un campo especifico de una consulta. Su sintaxis es:

```
Min(expr)
Max(expr)
```

Donde *expr* es el campo sobre el que se desea realizar el cálculo. *Expr* puede 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).

```
SELECT Min(Gastos) AS ElMin FROM Pedidos WHERE Pais = 'España';
SELECT Max(Gastos) AS ElMax FROM Pedidos WHERE Pais  = 'España';
```

Sum

Devuelve la suma del conjunto de valores contenido en un campo específico de una consulta. Su sintaxis es:

```
Sum(expr)
```

Donde *expr* representa 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).

```
SELECT Sum(PrecioUnidad * Cantidad) AS Total FROM DetallePedido;
```