Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

SQL. MANIPULACIÓN DE DATOS

INTRODUCCIÓN A SQL. CONSULTAS SIMPLES

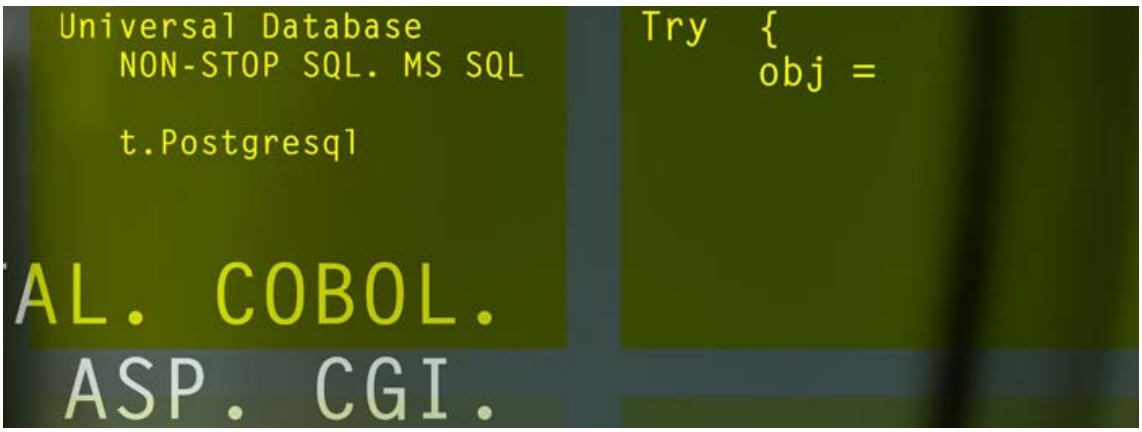
Índice

Presentación

Como resultado de la fase de diseño de la base de datos obtenemos el esquema relacional de la misma. Anteriormente aprendimos a obtener este esquema relacional. El siguiente paso sería refinarlo mediante el proceso de normalización que permite obtener el esquema más eficiente posible. El proceso de normalización lo veremos más adelante.

Ahora vamos aprender a **construir y manipular la BD**. Para ello necesitamos elegir cuál será el SGBD que utilizaremos y cuál es el lenguaje que éste utiliza tanto para la manipulación como para la definición de los datos. El lenguaje del SGBD más extendido es el **SQL**, llegando a convertirse en un estándar. Éste será el lenguaje que estudiaremos omitiendo el estudio de otros como QUEL, QBE ó ISBL por considerarlos obsoletos.

En esta unidad nos centraremos en el estudio del **sublenguaje para la manipulación de los datos (LMD)** que permite realizar consultas y manipular los datos almacenados. Y posteriormente estudiaremos el sublenguaje de definición de datos (LDD) que permite construir las tablas para definir el esquema de la BD.



Introducción a SQL

Anteriormente aprendimos que, en el modelo relacional, las operaciones se definen en base a modelos matemáticos como el algebra relacional, el cálculo relacional de tuplas (CRT) y el cálculo relacional de dominios (CRD). Vimos también que los lenguajes relacionales comerciales se basaban en éstos, **SQL es una combinación de todos**.

SQL fue desarrollado por IBM a finales de los 70 a partir del lenguaje SEQUEL. Las siglas de SQL provienen de *Structured Query Language* (Lenguaje estructurado de consultas). Fue estandarizado por el ANSI en 1986 obteniéndose como resultado el **SQL-86 o SQL-1**. En 1987 fue confirmada por ISO.
En 1992 apareció una nueva versión de manos de ANSI e ISO que incluía algunas funcionalidades que habían sido echadas en falta por los desarrolladores y suprimía otras relacionadas con el almacenamiento. Se obtuvo así el **SQL-92 ó SQL-2** esta es la versión que se utiliza como referencia en esta materia por ser el estándar al que da soporte la mayoría de los SGBD comerciales.

SQL-3, también llamado **SQL-1999,** apareció en 1999. Esta versión incluía expresiones regulares, consultas recursivas, triggers y características orientadas a objetos.

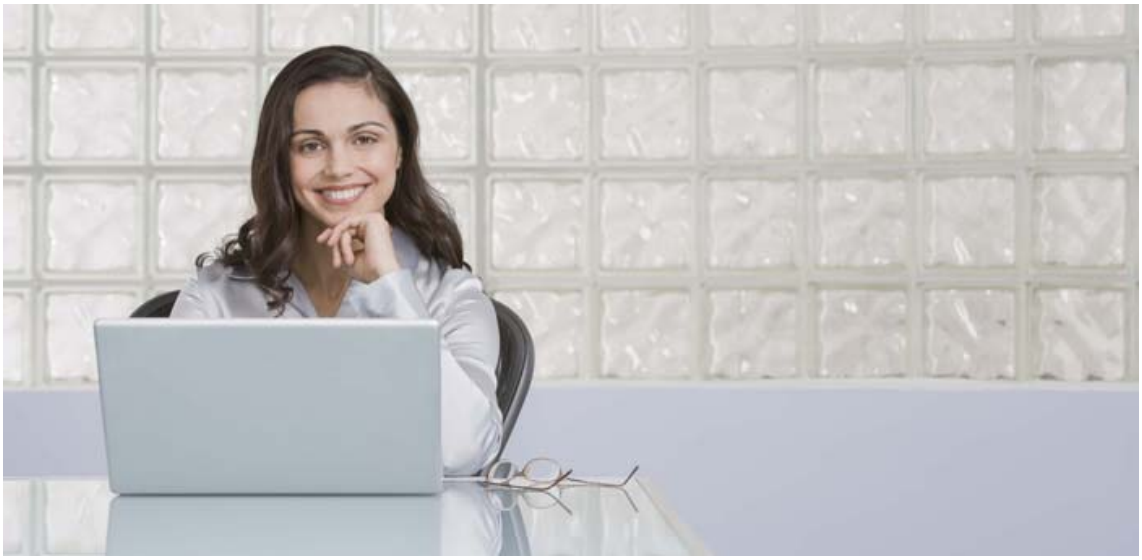
Componentes de SQL

SQL tiene varios componentes:

LDD (Lenguaje de Definición de Datos)	Permite construir y modificar el esquema de la base de datos. Incluye operaciones para definir esquemas de relaciones, borrar relaciones, modificar esquemas de relaciones, definir vistas...
LMD (Lenguaje de Manipulación de Datos)	Permite acceder y modificar la instancia de la base de datos. Incluye operaciones para realizar: • consultas, obteniendo así los datos necesarios; • inserciones, introduciendo nuevos datos en las relaciones; • eliminaciones, para borrar datos; y • modificaciones, para actualizar los datos de las tablas.
Control de transacciones	SQL incluye la sintaxis de las órdenes para definir el comienzo y la finalización de transacciones.
SQL incorporado o empotrado	Define cómo incorporar instrucciones de SQL en lenguajes de programación de propósito general (ADA, C++, Java, etc.). De esta forma se le permite al desarrollador de aplicaciones interactuar con el SGBD para lanzar operaciones, tanto del LMD como del LDD, sobre la BD.
Integridad	El LDD incluye órdenes para chequear la integridad de los datos almacenados.
Autorización	El LDD incluye órdenes para indicar permisos de acceso a relaciones y vistas.

Presentación de la BD bancaria

A lo largo de este tema vamos a aprender a realizar algunas consultas sencillas. Utilizaremos ejemplos que ilustren las sentencias estudiadas. La base de datos que vamos a utilizar en todos estos ejemplos es la BD bancaria que supondremos contiene la siguiente información.



Ejemplo **Clientes** Ejemplo **Depósitos** Ejemplo **Préstamos**

Ejemplo **Sucursales**

Ejemplo

CLIENTES (nombre_cli, calle, ciudad)

Almacena los clientes del banco.

nombre_cli	calle	ciudad
Alvarez	Castilla 12	Madrid
Gaya	Franco	Madrid
Gonzalez	Rio verde	Málaga
Gutierrez	Alcala 12	Granada
Lopez	Constitución	Granada
Perez	Alcala 12	Madrid
Pineda	Alcala 1	Málaga

Ejemplo

DEPÓSITOS (sucursal, num_cta, nombre_cli, saldo)

Almacena las cuentas del banco.

sucursal 1	num_cta	nombre_cli	saldo
Neptuno	1	Perez	1500
Caleta	2	Perez	11500
Caleta	3	Gaya	500
Nueva	4	Alvarez	100
Nueva	5	Pineda	15000
Zaidin	10	Perez	200
Zaidin	11	Alvarez	200

Ejemplo

PRÉSTAMOS (sucursal, n_prestamo, nombre_cli, cantidad)

Préstamos concedidos a los clientes y cuantía de los mismos.

sucursal	n_prestamo	nombre_cli	cantidad
Caleta	1	Perez	1000
Neptuno	2	Gaya	1000
Caleta	3	Gaya	1000
Nueva	4	Pineda	1000
Nueva	5	Gonzalez	1000
Zaidin	10	Alvarez	2300

Ejemplo

SUCURSALES (sucursal, activo, ciudad)

Sucursales de las que dispone el banco.

sucursal	activo	ciudad
Bacalao	3.50032e+006	Albolote
Caleta	300000	Granada
Neptuno	3000	Granada
Nueva	3e+006	Motril
Zaidin	3.00005e+007	Granada

Consultas simples. Selección de columnas

Lo primero que vamos a aprender es la sintaxis de la sentencia en SQL que permite consultar los datos actualmente almacenados. Es la **consulta SELECT** y tiene la siguiente sintaxis:

```
SELECT <atributos> FROM <relación> WHERE <condición>
```

Como resultado de una sentencia de este tipo obtendremos una tabla que será un subconjunto de la tabla especificada en el FROM (<relación>).Tendrá las columnas que se especifiquen en el SELECT (<atributos>) y las filas que cumplan la condición especificada en el WHERE (<condición>).
La consulta más sencilla que se puede realizar es aquella que muestra la tabla original completa:

```
SELECT * FROM <tabla>
```

Al poner * en la selección de atributos estamos indicando que queremos mostrarlos todos.

Seleccionar algunas de las columnas	**SELECT ciudad FROM cliente;**
Evitar repeticiones	**SELECT DISTINCT ciudad FROM cliente;**
Aplicar operaciones	**SELECT n_prestamo, cantidad*1.05 FROM prestamo;**
Renombrar una columna	**SELECT n_prestamo, cantidad*1.05 AS incremento FROM prestamo;**

Seleccionar algunas columnas

Para seleccionar algunas de las columnas sólo tendríamos que indicar los nombres de los atributos que queremos que se muestren. Por ejemplo, si quisiéremos saber las ciudades a las que pertenecen nuestros clientes para lanzar en estas una campaña publicitaria utilizaríamos la consulta SELECT ciudad FROM cliente. El resultado sería el que se muestra.

ciudad
Madrid
Madrid
Malaga
Granada
Granada
Madrid
Málaga

Evitar repeticiones

Para evitar que aparezcan repetidas las ciudades, incluimos la palabra clave DISTINCT antes de la lista de atributos a mostrar.

ciudad
Madrid
Málaga
Granada

Aplicar operaciones

Pueden aplicarse operaciones (+, -, *. /) a los datos consultados (campo calculado o derivado). La siguiente consulta muestra los préstamos y un incremento del 5% sobre los mismos que ha decidido aplicar el banco.

n_prestamo	cantidad*1.05
1	1050
2	1050
3	1050
4	1050
5	1050
10	2415

Renombrar una columna

Como se puede comprobar, como nombre de la columna en la tabla asociada, aparece la operación que hemos realizado. Si queremos modificar este valor tendríamos que utilizar un alias añadiendo la cláusula AS <nombre_columna> a continuación de la operación. SELECT n_prestamo, cantidad* 1.05 AS incremento.

n_prestamo	incremento
1	1050
2	1050
3	1050
4	1050
5	1050
10	2415

Consultas simples. Selección de filas

Para seleccionar las filas a mostrar en el resultado de una consulta tenemos que establecer la condición que debe cumplir cada instancia de la relación para ser seleccionada. Por ejemplo si quisiéramos saber los clientes de nuestro banco que tienen una cuenta con un saldo superior a 10000 para avisarles de nuestros fondos de inversión, tendríamos que utilizar la consulta:

```
SELECT nombre_cli FROM depósitos WHERE saldo > 10000
```

nombre_cli
Perez
Pineda

Existen cinco **condiciones de búsqueda básicas** en terminología ISO que se pueden añadir a continuación de la cláusula where:

Comparación entre valores	En SQL pueden usarse los siguientes operadores de comparación: =, <>, <, >, >=, <=, != (distinto está permitido en algunos dialectos). Pueden crearse comparaciones más complejas utilizando operadores lógicos: AND, OR y NOT. Siempre es recomendable el uso de paréntesis para indicar el orden de evaluación de las expresiones y evitar ambigüedades.
Rango	Comprueba si una expresión está en un rango dado. BETWEEN/ NOT BETWEEN
Pertenencia a un conjunto	Comprueba si el valor de una expresión está dentro de los pertenecientes a un conjunto dado. IN/ NOT IN
Correspondencia de patrones	Comprueba si una cadena de caracteres se ajusta a un patrón dado. LIKE/ NOT LIKE
Nulo	Comprueba si una columna tiene un valor nulo. IS NULL/ IS NOT NULL.

Consultas simples. Comparación, rango y pertenencia a un conjunto

Comparación

Veamos la selección de filas utilizando comparación con un ejemplo: queremos saber cuáles de las cuentas de PEREZ tienen un saldo superior a 1000 y no son de la sucursal NEPTUNO. Para realizar esta consulta hay que crear una condición triple: seleccionar las filas de depósitos cuyo nombre de cliente sea PEREZ, cuyo saldo sea superior a 1000 y cuya sucursal no sea NEPTUNO. La consulta sería:

```
SELECT * FROM depositos WHERE nombre_cli='PEREZ' AND saldo>1000 AND
(NOT (sucursal='NEPTUNO'));
```

Rango

Se puede establecer la condición indicando el rango de valores a seleccionar. Para ello se utiliza la cláusula BETWEEN y NOT BETWEEN. Supongamos, por ejemplo, que queremos saber cuáles son las cuentas con un saldo entre 1000 y 10000:

```
SELECT * FROM depositos WHERE saldo BETWEEN 1000 AND 10000
```

Pertenencia a un conjunto

En este caso la condición la establecemos indicando el conjunto de los valores que puede aparecer en la columna para seleccionar esta fila como resultado de la consulta. Por ejemplo para saber cuáles son los clientes andaluces utilizaríamos la consulta:

```
SELECT * FROM cliente WHERE ciudad IN
('GRANADA','MALAGA','CADIZ','HUELVA','SEVILLA','CORDOBA','JAEN','ALMERIA')
```

Correspondencias simples. Correspondencia de patrones y nulos

Selección basada en la correspondencia de patrones

Podemos establecer la condición para seleccionar las filas indicando el formato ó patrón que tiene la cadena de caracteres que aparece como valor en una determinada columna. Para ello utilizamos las cláusulas LIKE y NOT LIKE. Para especificar el patrón se utilizan expresiones que pueden incluir los caracteres:

- % indica cualquier cadena de 0 o más caracteres
- ? indica cualquier carácter individual

Como ejemplo, localizar todos los clientes en cuya dirección figure el texto Alcalá:

```
SELECT * FROM cliente WHERE calle LIKE %ALCALA%;
```

Tabla resultado

En detalle

nombre_cli	calle	ciudad
Gutierrez	Alcala 12	Granada
Perez	Alcala 12	Madrid
Pineda	Alcala 1	Malaga

Algunos ejemplos de patrones podrían ser los siguientes.

Ejemplos de patrones

- • ‘Guada%’ encaja con cualquier cadena que comience con Guada: Guadarrama, Guadalajara, Guadalquivir, Guadalcanal, …

  •’ %illa%’ en caja con cualquier cadena que contenga illa: Jarandilla de la Vera, Villanueva, Villalba, …

  • ‘_ _ _’ encaja con cualquier cadena que SÓLO tenga 3 caracteres.

  • ‘_ _ _ %’ encaja con cualquier cadena de AL MENOS 3 caracteres.

Selección basada en campos nulos (NULL)

Cuando se insertan datos en una tabla existe la posibilidad de no especificar el valor de algunas columnas. Por ejemplo, puedo introducir un nuevo cliente en nuestra base de datos dejando en blanco la calle porque no la conozca en este momento. El SGBD automáticamente asignará un valor NULL a este campo.

La selección basada en campos nulos consiste en seleccionar las filas que tienen un valor nulo en la columna especificada.

"Localizar todos los clientes sin dirección especificada". Es de prever que esta consulta aplicada a nuestra BD de como resultado una tabla vacía puesto que hemos establecido el valor de las calles de todos nuestros clientes:

```
SELECT * FROM cliente WHERE calle IS NULL;
```

nombre_cli	calle	ciudad

Consultas simples. Ordenación de resultados

Tras el WHERE de una consulta SELECT se puede utilizar la cláusula ORDER BY para indicar que queremos que se nos muestren los resultados ordenados por una determinada columna. La consulta genérica final quedaría:

```
SELECT <atributos> FROM <relación> WHERE <condición> ORDER BY <atributos>;
```

El atributo por el cual se realiza la ordenación se llama **clave primaria de ordenación**. Si la clave primaria de ordenación tiene valores repetidos, puede ordenarse según una clave secundaria.

Por defecto se ordena de forma ascendente (**ASC**), pero puede indicarse que sea descendente (**DESC**). Si quisiéramos mostrar las cuentas de la tabla depósitos ordenadas primero por sucursal y después por el nombre del cliente tendríamos que realizar la consulta:

```
SELECT * FROM depositos ORDER BY sucursal ASC, nombre_cli DESC;
```

sucursal	num_cta	nombre_cli	saldo
Caleta	2	Perez	11500
Caleta	3	Gaya	500
Neptuno	1	Perez	1500
Nueva	5	Pineda	15000
Nueva	4	Alvarez	100
Zaidin	10	Perez	200
Zaidin	11	Alvarez	200

Consultas multitabla simples

Supongamos ahora que nuestro banco quiere enviar cartas de agradecimiento a los clientes más ahorradores. Para ello necesitamos el nombre, la calle y la ciudad de todos los clientes que tengan cuentas con saldo superior a los 10.000 euros. Para conseguir esta consulta necesitaríamos una tabla que tuviese el nombre, la calle, la ciudad y el saldo de las cuentas de todos los clientes con depósitos. Tal tabla no existe en nuestra base de datos. Sin embargo, si pudiésemos fundir las tablas de clientes (con el nombre, la calle y la ciudad) y la de depósitos (con el número de cuenta y el saldo) podríamos obtener la tabla que necesitamos y establecer sobre esta las condiciones para la selección de filas (tener alguna cuenta con saldo superior a 10.000 euros). Esto se consigue realizando una consulta multitabla. Para crear una consulta multitabla basta con indicar las tablas, separadas por comas, tras la cláusula FROM. Puede indicarse un alias para cada tabla tras el nombre de la misma. De esta forma, el resultado de la consulta anterior sería:

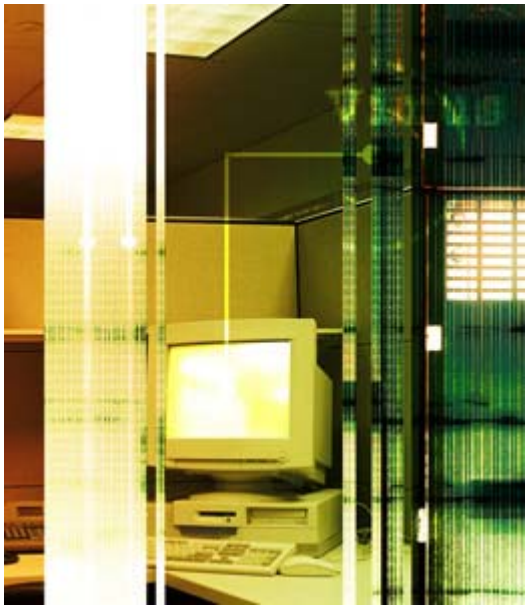
```
SELECT c.nombre_cli, calle, ciudad FROM cliente c, depositos d
WHERE c.nombre_cli=d.nombre_cli AND d.saldo>10000;
```

nombe_cli	calle	ciudad
Perez	Alcala 12	Madrid
Pineda	Alcala 1	Málaga

Analicemos la consulta anterior:

- `cliente c, depositos d:` Establecemos cuáles serán las tablas que queremos fusionar y utilizamos alias: c para clientes y d para depósitos, de esta forma podemos hacer referencia a las mismas tanto con el alias como con el nombre de la tabla.
- `c.nombre_cli=d.nombre_cli.` Si no indicamos ninguna condición en la cláusula where el resultado es el producto cartesiano de clientes y depósitos, es decir una tabla en la que cada fila de la tabla depósitos se relaciona con cada fila de la tabla clientes. De esta forma para la cuenta 1 aparecerían 7 filas, una para cada uno de los clientes de nuestra base de datos. Nosotros queremos solo los datos del cliente que es el propietario de la cuenta actual por lo que establecemos esta condición. Nótese que como nombre_cli aparece tanto en depósitos como en clientes es necesario indicar a cuál de estas columnas nos referimos utilizando el nombre de la tabla (o su alias correspondiente) previo al nombre del atributo.

Resumen



SQL es el **lenguaje del SGBD estándar más utilizado** en la actualidad.

SQL permite **consultar datos y modificar la instancia** de la base de datos a través de su LMD. También permite crear y modificar el esquema de la base de datos, chequear la integridad de la misma y definir las autorizaciones a través de su LDD.

Para consultar los datos se utiliza la sentencia **SELECT.** Está compuesta por varias partes:

- SELECT donde se indican las columnas que se quieren seleccionar en la tabla resultado,
- FROM para indicar las tablas de donde se quiere extraer la información,
- WHERE donde se establece la condición para la selección de las filas, y
- ORDER BY que permite indicar los atributos por los que se quiere ordenar el resultado.

La condición del WHERE se puede establecer en base a: **comparaciones** (con los operadores de comparación y los operadores lógicos), **rangos** (con la cláusula BETWEEN), **pertenencias a conjuntos** (con la cláusula IN), **correspondencia de patrones** (con la cláusula LIKE) y **comprobación de valores nulos** (con la cláusula IS NULL).