Universidad
Europea de Madrid

LAUREATE INTERNATIONAL UNIVERSITIES

SQL. MANIPULACIÓN DE DATOS

SQL. CONSULTAS AVANZADAS

Índice

Presentación

Ya sabemos realizar consultas que permiten selección de columnas, selección de filas, ordenación y consultas multitabla sencillas. Ahora aprenderemos a realizar consultas mucho más potentes sobre la base de datos que nos permitan:

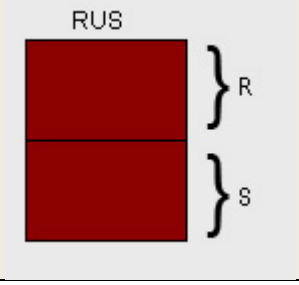
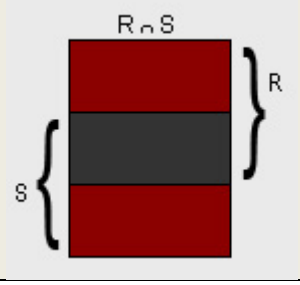
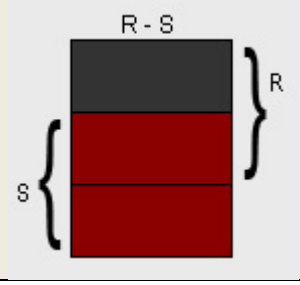
- Utilizar operaciones de conjuntos: **combinación de tablas** de resultados.
- Utilizar una consulta dentro de otra: creación de **subconsultas**.
- Aplicar funciones (suma, promedio, contar el número de filas, etc.) a una tabla: uso de **funciones de agregación.**
- Agrupar las filas de una tabla en base a un atributo: **agrupaciones**.
- Establecer **condiciones** a los grupos previamente creados para indicar si queremos que se muestren o no.
- Utilizar como condición de una consulta que el resultado de otra sea el conjunto vacío o no.

Con este conjunto de operaciones podemos crear consultas para solucionar cualquier tipo de necesidad que se haya establecido en la especificación de requisitos del SBD a realizar.

Introducción a SQL

Siguiendo con nuestra base de datos bancaria, supongamos que ahora necesitamos obtener un listado con todos los clientes de una determinada sucursal de nuestro banco, tanto aquellos que tienen cuenta como aquellos que tienen un préstamo en la misma. Para solucionar esta consulta necesitamos dos tablas: préstamos y depósitos, pero no queremos fundir las dos tablas en una, como en el caso de las consultas multitabla, sino extraer de préstamos los nombres de los clientes con un préstamo en nuestro banco y añadirles los clientes que aparecen en la tabla depósitos (tienen una cuenta en la misma). **Necesitamos combinar la tabla resultado de obtener los clientes con préstamo y la de obtener los clientes con cuenta.**

En SQL pueden usarse las operaciones de combinación de conjuntos UNION, INTERSECCIÓN y DIFERENCIA para mostrar una única tabla de resultado. Para ello el estándar ISO proporciona los operadores UNION, INTERSECT y EXCEPT (MINUS en algunos dialectos de SQL).

UNION	INTERSECT	EXCEPT
		


```
(SELECT nombre_cli FROM depositos) UNION (SELECT nombre_cli FROM
prestamo);
```

El resultado de nuestra consulta **sería**:

Tabla

nombe_cli
Perez
Gaya
Alvarez
Pineda
Gonzalez

Más ejemplos

Ejemplo

Listado de los clientes que tienen cuenta y préstamo en nuestro banco.

```
(SELECT nombre_cli FROM depositos) INTERSECT (SELECT
nombre_cli FROM prestamo);
```

Listado de clientes que tienen cuenta pero no tienen préstamo.

```
(SELECT nombre_cli FROM depositos) EXCEPT (SELECT
nombre_cli FROM prestamo);
```

Subconsultas

El banco ha decidido regalar a sus clientes más cercanos una batería de cocina. Para ello necesita saber cuáles son los clientes que viven en alguna de las ciudades donde tienen sucursal. Si consiguiéramos obtener el conjunto de ciudades en las que nuestro banco tiene sucursal sólo tendríamos que ver si la ciudad del cliente se encuentra dentro de este conjunto, cláusula IN. Necesitamos realizar una subconsulta que dé como resultado las ciudades del banco.

Una subconsulta consiste en realizar una consulta SELECT dentro de otra. Existen 3 tipos de subconsultas:

- **Escalar**: devuelve un único valor. Inicialmente puede usarse en cualquier lugar donde haga falta un único valor.
- **De fila**: devuelve múltiples columnas de una única fila.
- **De tabla**: devuelve múltiples columnas y múltiples filas.
-

Las dos últimas pueden usarse en las cláusulas WHERE y HAVING y también en las instrucciones INSERT, UPDATE y DELETE. Sentencias del SQL que permiten modificar la instancia de la BD y que estudiaremos en la próxima lección.
La consulta sería:

```
SELECT * FROM cliente WHERE ciudad IN (SELECT ciudad FROM
sucursal);
```

nombe_cli	calle	ciudad
Gutierrez	Alcala 12	Granada
Lopez	Constitución	Granada

Subconsultas. Cláusulas ANY y ALL

Las cláusulas ANY y ALL se usan para construir condiciones basadas en subconsultas.

- Si la condición utiliza ALL, sólo valdrá VERDADERO si se satisface para TODOS los valores de la subcosulta.
- Si la condición utiliza ANY, valdrá VERDADERO si se satisface para ALGUNO de los valores obtenidos por la subconsulta.

Si la subconsulta está vacía, la condición ALL devuelve VERDADERO, pero ANY devolverá FALSO. Según qué estándar, puede usarse SOME en lugar de ANY.

Clientes con cuentas cuyo saldo sea mayor a todos los de la sucursal CALETA:

```
SELECT nombre_cli FROM depositos WHERE saldo >ALL (SELECT saldo
FROM depositos WHERE sucursal='CALETA');
```

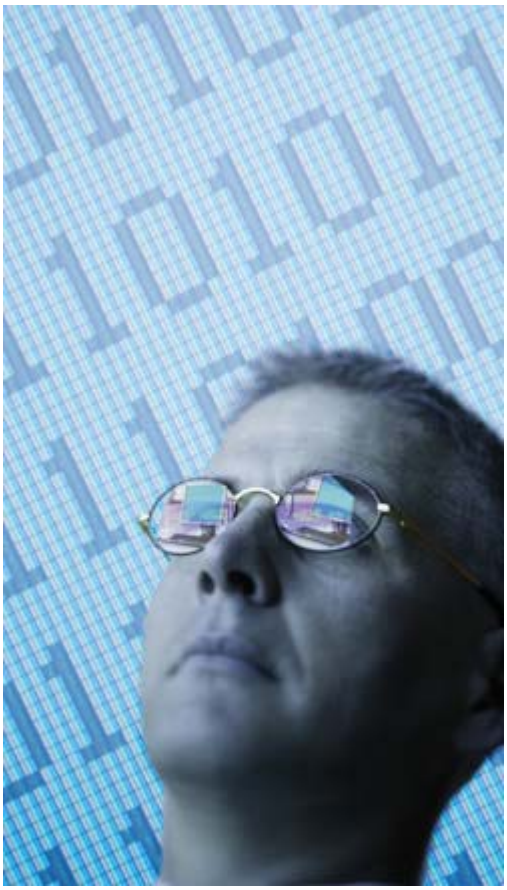
nombe_cli
Pineda

Clientes con cuentas cuyo saldo sea mayor a alguno de los de la sucursal CALETA:

```
SELECT DISTINCT nombre_cli FROM depositos WHERE saldo
>ANY(SELECT saldo FROM depositos WHERE sucursal='CALETA');
```

nombe_cli
Pérez
Pineda

Funciones de agregación



En algunas ocasiones no necesitamos obtener como resultado de una consulta una tabla resultado sino un valor que nos indique, por ejemplo, el número de filas de una tabla, la media, valor más grande ó más pequeño de los que aparecen en una columna, etc. En estos casos necesitamos utilizar funciones de agregación.

El estándar ISO define 5 **funciones de agregación**, también llamadas de resumen:

- **COUNT**. Devuelve el número de valores de un atributo dado.
- **SUM**. Devuelve la suma de los valores de un atributo dado.
- **AVG**. Indica la media de los valores contenidos en una columna dada.
- **MIN**. Devuelve el valor más pequeño contenido en una columna.
- **MAX**. Indica el valor máximo contenido en una columna dada.

Las funciones de agregación operan sobre una única columna y devuelven un único valor. Las **restricciones de uso** son:

- COUNT, MIN, MAX: operan sobre campos numéricos y no numéricos.
- AVG y SUM : operan SOLO sobre campos numéricos.
- Todos eliminan valores nulos antes de operar.

COUNT(*) es un caso especial. Cuenta todas las filas, incluyendo nulos y duplicados de la tabla resultado de la consulta. Si necesitamos no contar los duplicados tenemos que usar antes DISTINCT.

Documentos **Funciones de agregación. Ejemplos**

Agrupación de resultados

El banco ha decidido lanzar una campaña de publicidad local en aquellas ciudades con más número de clientes; necesita, por tanto, saber el número de clientes en cada ciudad. Lo que necesitamos es una tabla con el nombre de la ciudad y el número de clientes en cada una de ellas. Con las herramientas que tenemos hasta ahora no podemos solucionar esta consulta. La cláusula GROUP BY agrupa los datos de las tablas indicadas mediante el SELECT, generando una única fila resumen para cada grupo. Las columnas (atributos) indicados tras el GROUP BY son llamadas columnas de agrupamiento y son las que indican sobre qué campos se realizarán los grupos. Se utiliza siempre después de la cláusula WHERE en caso de haberla. En el ejemplo inicial nosotros necesitamos agrupar por ciudades, por tanto, será ciudad de la tabla clientes el campo de agrupamiento. La sentencia SELECT completa quedaría:

```
SELECT <atributos> FROM <relación> WHERE <condición> GROUP BY
<campo_de_agrupamiento>;
```

La solución de la consulta sería:

```
SELECT ciudad, COUNT(DISTINCT nombre_cli) AS n_clientes FROM
cliente GROUP BY ciudad;
```

Cuando se usa GROUP BY, cada columna indicada tras el SELECT debe tener UN UNICO valor para cada grupo. Además, sólo pueden aparecer:

- Nombres de atributos (columnas),
- Funciones de agregación,
- Constantes,
- Combinaciones de los anteriores elementos.

Tabla Resultado

Ejemplos

Tabla Resultado

ciudad	n_clientes
Granada	2
Madrid	3
Málaga	2

Ejemplos

Calcular el saldo medio de las cuentas de cada sucursal.

```
SELECT sucursal, AVG(saldo) AS saldo_medio FROM depositos GROUP
BY sucursal;
```

sucursal	saldo_medio
Caleta	6000
Neptuno	1500
Nueva	7550
Zaidin	200

Encontrar el número de clientes distintos que tienen cuenta en cada sucursal.

```
SELECT sucursal, COUNT(nombre_cli) AS n_clientes FROM depositos
GROUP BY sucursal;
```

sucursal	n_clientes
Caleta	2
Neptuno	1
Nueva	2
Zaidin	2

Selección de grupos

De la misma forma que se seleccionan filas en la cláusula WHERE, se pueden seleccionar los grupos a mostrar en la tabla resultado utilizando la cláusula HAVING junto con la condición que tienen que cumplir los grupos para ser mostrados. Las columnas indicadas en HAVING deben aparecer antes en algún lugar de la consulta.

Encontrar las sucursales cuyo saldo medio supere los 1200 euros

```
SELECT sucursal, AVG(saldo) AS saldo_medio FROM depositos GROUP
BY sucursal HAVING saldo_medio>1200;
```

sucursal	saldo_medio
Caleta	6000
Neptuno	1500
Nueva	7550

Encontrar la sucursal con el mayor saldo medio

```
SELECT sucursal FROM depositos GROUP BY sucursal HAVING
AVG(saldo)>=ALL (SELECT AVG(saldo) FROM depositos GROUP BY
sucursal);
```

Encontrar los nombres de los clientes y la diferencia entre saldo en cuentas y el total prestado

Ejemplo

Podemos ver la explicación **aquí**.

Ejemplo

```
SELECT c.nombre_cli, total_en_cuentas-total_prestado AS
saldo_real FROM

(SELECT nombre_cli, SUM(saldo)AS total_en_cuentas FROM
depositos GROUP BY nombre_cli) AS c,

(SELECT nombre_cli, SUM(cantidad) AS total_prestado FROM
prestamo GROUP BY nombre_cli) AS p

WHERE c.nombre_cli=p.nombre_cli;
```

Explicación

1. Creo dos tablas temporales:

Una que contiene los nombres de los clientes y la suma del total de sus cuentas que llamo c:

```
SELECT nombre_cli, SUM(saldo) FROM depositos GROUP BY
nombre_cli;
```

Otra con la suma de las cantidades de los préstamos concedidos por cliente que llamo p.

```
SELECT nombre_cli, SUM(cantidad) FROM prestamo GROUP BY
nombre_cli;
```

2. Fusiono ambas tablas mediante una consulta multitabla para obtener el nombre del cliente, y la diferencia entre saldo en cuentas y el total prestado.

```
SELECT nombre_cli, SUM(saldo) FROM depositos GROUP BY
nombre_cli;
```

nombe_cli	saldo_real
Alvarez	-2000
Gaya	-1500
Perez	12200
Pineda	14000

Exists y not exists

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 EXISTS <subconsulta>;
```

El resultado de EXISTS <subconsulta> será verdadero si la tabla resultado de la subconsulta contiene alguna fila y será falso si está vacía. NOT EXISTS es el contrario.

Esta cláusula se utiliza mucho en casos similares a este: queremos conocer los clientes que tienen cuenta en TODAS las sucursales del banco en Granada. Esta consulta podría reformularse de la siguiente manera: nombres de clientes de la tabla clientes tales que no exista una sucursal de Granada con la que el cliente no esté relacionado en la tabla de depósitos.

```
SELECT nombre_cli FROM cliente c WHERE NOT EXISTS
    (SELECT sucursal FROM sucursal s WHERE ciudad='GRANADA' AND
NOT EXISTS
        (SELECT * FROM depositos d WHERE
c.nombre_cli=d.nombre_cli AND

 s.sucursal=d.sucursal));
```

nombre_cli
Perez

Resumen

Las características avanzadas que se pueden utilizar en la sentencia SELECT de SQL permiten: combinar tablas resultado, realizar subconsultas, utilizar funciones de agregación, agrupar resultados.

Las cláusulas que se utilizan para **combinar tablas resultado** son: **UNION**, **INTERSECT** y **EXCEPT**. Se utilizan para unir dos consultas SELECT y el resultado se obtiene de aplicar la operación de conjuntos.

Una **subconsulta** es una consulta SELECT dentro de otra. Puede aparecer:

- en el SELECT si el resultado es un escalar,
- en el FROM para crear tablas temporales,
- en el WHERE ó en el HAVING para establecer la condición. En éste último caso se utiliza junto con los operadores de comparación ó las cláusulas **ANY**, **ALL** y **EXISTS**.

La **agrupación de resultados** permite obtener una fila resumen para todas aquellas que coincidan en valor en una columna. Se crean con la cláusula **GROUP BY**. Se pueden establecer condiciones a los grupos utilizando la cláusula **HAVING**.

Las **funciones de agregación** permiten calcular valores sobre las filas ó columnas de las tablas ó sobre los grupos creados. Son: **SUM, AVG, MIN, MAX, COUNT**.