



**Universidad
Europea de Madrid**

LAUREATE INTERNATIONAL UNIVERSITIES

SQL. MANIPULACIÓN DE DATOS

EJERCICIO RESUELTO

Índice

| | |
|--|----|
| Presentación..... | 3 |
| Consultas simples de selección de filas | 4 |
| Consultas complejas con combinación de tablas de resultado | 7 |
| Consultas complejas con subconsultas | 9 |
| Consultas complejas con funciones de agregación | 10 |
| Consultas complejas con agrupación de resultados I..... | 12 |
| Consultas complejas con agrupación de resultados II..... | 13 |
| Inserción de datos | 15 |
| Modificación de datos | 16 |
| Eliminación de datos | 18 |
| Resumen..... | 19 |

Presentación

A través de los tres temas anteriores de esta unidad hemos estudiado cómo realizar **consultas simples y complejas** y cómo **modificar la instancia de la base de datos** para **introducir, modificar y borrar** filas de las relaciones.

Para un mejor seguimiento de las consultas correspondientes puedes utilizar el recurso *Bases de datos de aerolíneas* para comprobar los resultados que se muestran. Se trata de una hoja Excel con el contenido de la base de datos de las aerolíneas donde cada hoja tiene una de las tablas, y puedes encontrarla en la carpeta de materiales del tema.



En este tema el alumno conseguirá:

- Aplicar estos conocimientos en un caso práctico, el **problema de las aerolíneas**, cuyo diagrama E/R y correspondiente esquema obtuvimos en el tema 4 de la unidad 2.
- Realizar ejemplos de las **sentencias SQL** del **LMD** estudiado en esta unidad utilizando esta base de datos.

Consultas simples de selección de filas

Para comprender mejor los conocimientos adquiridos hasta ahora, vamos a realizar unos ejemplos de consultas simples.

Ejemplo 1

Ejemplo 2

Ejemplo 3

Ejemplo 4

Códigos de los vuelos de los que se compone la reserva de código 2

La tabla **compuesta_por** contiene los vuelos de los que está compuesta cada reserva. Tenemos que consultar esta tabla y mostrar los **códigos de vuelo** de las filas correspondientes a la reserva de **código 2**. Se trata de una sentencia **SELECT** sencilla donde la condición está compuesta por una comparación.

```
SELECT cod_vuelo FROM compuesta_por WHERE cod_reserva=2;
```

Ejemplo 1

Ejemplo 2

Ejemplo 3

Ejemplo 4

Vuelos planificados con una duración entre 5 y 15 horas y que no salgan de la T4.

Se trata de una consulta simple de selección de filas de la tabla **vuelos** donde la condición contiene una **comparación** y un **rango**.

```
SELECT * FROM vuelos WHERE duracion BETWEEN 5 AND 15 AND NOT(salida_terminal='T4');
```

Ejemplo 1

Ejemplo 2

Ejemplo 3

Ejemplo 4

Clientes cuyo correo electrónico sea de Hotmail

Tenemos que buscar en la **tabla de clientes** aquellos cuyo **correo_e** contenga la subcadena **'hotmail'**. Hay que utilizar **correspondencia de patrones**.

```
SELECT * FROM cliente WHERE correo_e LIKE '%hotmail%';
```

Ejemplo 1

Ejemplo 2

Ejemplo 3

Ejemplo 4



Cientes con tarjetas de crédito 6000 ó 4B. Ordenar los resultados por titular de forma descendente.

Es una consulta de selección de filas de la tabla **tarjeta_credito** donde la condición se establece por pertenencia a un conjunto. Hay que utilizar la cláusula **ORDER BY** por titular.

```
SELECT * FROM tarjeta_credito WHERE tipo IN ('4b','6000') ORDER BY titular DESC;
```



En detalle

Resultado 1



Documentos

[Resultado 2](#)

En detalle

[Resultado 3](#)

En detalle

[Resultado 4](#)

Resultado 1

| <u>cod_vuelo</u> |
|------------------|
| <u>564</u> |
| <u>598</u> |
| <u>992</u> |

Resultado 3

| correo_e | telefono |
|----------------------|-----------|
| alberca@hotmail.com | 912115339 |
| albero@hotmail.com | 912115224 |
| aliane@hotmail.com | 912115248 |
| aliberti@hotmail.com | 912113502 |
| alonso@hotmail.com | 912115214 |
| alvarez@hotmail.com | 912115205 |
| aparicio@hotmail.com | 912115309 |
| araez@hotmail.com | 912113676 |
| aragones@hotmail.com | 912115376 |
| arias@hotmail.com | 912115373 |

Resultado 4

| n_tarjeta | tipo | caducidad_mes | caducidad_ano | titular | pais | cod_seguridad |
|-----------|------|---------------|---------------|----------|----------|---------------|
| 3 | 6000 | 5 | 2014 | alvarez | alemania | 02 |
| 1 | 4b | 3 | 2012 | aliberti | españa | 00 |
| 5 | 4b | 7 | 2016 | albero | españa | 04 |
| 9 | 4b | 11 | 2020 | alberca | españa | 08 |

Consultas complejas con combinación de tablas de resultado

Seguimos con los ejemplos, en este caso consultas complejas que requieren combinar varias tablas.

Nombres de los pasajeros que llevan a su cargo un bebé y que hayan pedido una dieta vegetariana

Los adultos que llevan a su cargo un bebé son los que están en la tabla **adulto_con_bebe**, el tipo de dieta que elige cada pasajero está en la tabla **aparece**. Hay que mostrar los **DNI** de una tabla, los DNI de la otra y unirlos.

```
(SELECT dni FROM adultos_con_bebe) UNION (SELECT dni FROM aparece  
WHERE comida_especial='vegetariana');
```



Resultado

| <u>Resultado</u> |
|------------------|
| dni |
| 5205 |
| 5214 |
| 5224 |
| 5248 |
| 5288 |
| 3502 |
| 5376 |

DNI de los pasajeros que no sean niños

Obtenemos el conjunto de todos los **DNI** de los **pasajeros** y le quitamos el conjunto de los DNI de los **niños**.

```
(SELECT dni FROM pasajeros) EXCEPT (SELECT dni FROM nino);
```

DNI de los pasajeros que toman dieta musulmana y son niños

Hay que buscar los DNI en las tablas **nino** y **aparece**.

```
(SELECT dni FROM nino) INTERSECT (SELECT dni FROM aparece WHERE  
comida_especial='musulmana');
```



Resultado

| Resultado | |
|-----------|------|
| | dni |
| | 5339 |

Consultas complejas con subconsultas

Seguimos con más ejemplos de consultas complejas.

Clientes cuya tarjeta de crédito caduque el año 2012

Necesitamos un listado con los números de teléfono de los clientes que para pagar una reserva hayan utilizado una tarjeta de crédito cuyo año de caducidad sea el 2012. Vamos utilizar una **tabla temporal** con los **correos electrónicos**, **teléfonos**, **números de tarjeta** y las **fechas de caducidad** de las mismas. Después, seleccionamos de esta tabla las filas de las **tarjetas que caducan el año 2012**.

```
SELECT * FROM
  (SELECT c.correo_e, c.telefono, t.n_tarjeta, t.caducidad_ano FROM
  cliente c,
    reserva r, tarjeta_credito t      WHERE c.correo_e=r.correo_e AND
    t.n_tarjeta=r.n_tarjeta) AS temp  WHERE temp.caducidad_ano=2012;
```

ResultadoResultado

| correo_e | telefono | n_tarjeta | caducidad_ano |
|---------------------|-----------|-----------|---------------|
| alvarez@hotmail.com | 912115205 | 1 | 2012 |

Hay otra forma mucho más sencilla de realizar la consulta, ¿sabes cuál es?

¿sabes cuál es?

Otra forma mucho más sencilla de realizar la consulta es:

```
SELECT c.correo_e, c.telefono, t.n_tarjeta, t.caducidad_ano FROM
cliente c, reserva r, tarjeta_credito t WHERE c.correo_e=r.correo_e
AND t.n_tarjeta=r.n_tarjeta AND t.caducidad_ano=2012;
```

Reservas que tengan el precio más caro

Creamos una subconsulta para que nos muestre los **precios** de todas las **reservas** y seleccionamos aquellas cuyo precio sea **mayor o igual** que todas las anteriores.

```
SELECT * FROM reserva WHERE precio_total >= ALL (SELECT precio_total
FROM reserva);
```

Resultado

Consultas complejas con funciones de agregación

Número de veces que algún adulto ha volado con nuestra aerolínea

La tabla **líneas** contiene la información del **número** y **tipo** de pasajeros. Hay que seleccionar de esta tabla las líneas de **adultos** y sumar el número de **pasajeros** de éstas.

```
SELECT SUM(n_pasajeros) AS n_adultos FROM lineas WHERE
concepto='adulto';
```


[Resultado](#)

Resultado

| n_adultos |
|-----------|
| 20 |

Precio medio de las reservas realizadas con destino a Atlanta

Esta información aparece en la tabla **reserva**.

```
SELECT AVG(precio_total) AS precio_medio FROM reserva WHERE
destino='atlanta';
```


[Resultado](#)

Resultado

| precio_medio |
|------------------|
| 1051.47500610352 |

Número de reservas para vuelos de la empresa Easy-jet y duración media de los mismos

En la tabla de **vuelos** aparece el nombre de la empresa. Hay que comprobar qué vuelos realiza esta empresa en la tabla **compuesta_por** y contar el **número de reservas** que se realizan sobre estos.

```
SELECT COUNT(DISTINCT cod_reserva) AS n_reservas_easyjet FROM vuelos
v, compuesta_por c WHERE v.cod_vuelo=c.cod_vuelo AND
v.empresa='easyjet';
```

[Resultado](#)**Resultado****n_reservas_easyjet**

4

Consultas complejas con agrupación de resultados I**Número de vuelos fletados por empresa**

En este caso tenemos que calcular para cada empresa que realiza vuelos una **fila resumen** que contenga el **nombre de la empresa** y el **número de vuelos** que realiza esta. Para ello agrupamos por empresa y contamos los vuelos para cada una, en la tabla **vuelos**.

```
SELECT empresa, COUNT(*) AS n_vuelos FROM vuelos v GROUP BY v.empresa;
```

Resultado

| empresa | n_vuelos |
|---------|----------|
| easyjet | 4 |
| iberia | 4 |
| kh | 4 |
| spanair | 3 |

Ingresos por empresa

Tenemos que calcular la suma de **precios totales** de las reservas realizadas sobre vuelos y agrupar por empresa. Utilizamos las tablas **reserva** y **vuelos**.

```
SELECT empresa, SUM(precio_total) AS ingresos FROM reserva r,  
compuesta_por c, vuelos v WHERE r.cod_reserva=c.cod_reserva AND  
c.cod_vuelo=v.cod_vuelo GROUP BY v.empresa;
```

**Resultado****Resultado**

| empresa | ingresos |
|---------|------------------|
| easyjet | 2427.40002441406 |
| iberia | 1613.5 |
| kh | 5091.5 |
| spanair | 9258 |

Consultas complejas con agrupación de resultados II

Empresa que menos ingresos ha realizado

Esta consulta es la misma que la anterior solo que ahora tenemos que establecer una condición a cada grupo utilizando la cláusula **HAVING**. En este caso la condición es que los ingresos sean menores que todos. Para ello utilizamos una subconsulta con los **ingresos** de todas las **empresas**, de forma similar a como hemos realizado previamente. Utilizamos las tablas **reserva**, **compuesta_por** y **vuelos**.

```
SELECT empresa, SUM(precio_total) AS ingresos
FROM reserva r, compuesta_por c, vuelos v
WHERE r.cod_reserva=c.cod_reserva AND c.cod_vuelo=v.cod_vuelo
GROUP BY v.empresa
HAVING ingresos <= ALL
    (SELECT SUM(precio_total) FROM reserva r, compuesta_por c, vuelos v
     WHERE r.cod_reserva=c.cod_reserva AND c.cod_vuelo=v.cod_vuelo
     GROUP BY v.empresa);
```


[Resultado](#)

Listado con las reservas y los precios calculados en base a la descripción de las líneas

El **precio** de la reserva será la **tarifa**, más las **tasas**, más los **gastos de gestión** por el **número de pasajeros** de ese concepto. Finalmente habrá que sumar estos valores para cada **reserva**.

```
SELECT cod_reserva, SUM((tarifa+tarifa*tasas+gastos_gestion)
*n_pasajeros) AS precio_calculado FROM lineas GROUP BY cod_reserva;
```


[Resultado](#)

Resultado

| cod_reserva | precio_calculado |
|-------------|------------------|
| 1 | 349 |
| 2 | 3870 |
| 3 | 6434 |
| 4 | 744 |
| 5 | 128995 |
| 6 | 7389 |
| 7 | 4926 |
| 8 | 5169 |
| 9 | 12891 |
| 10 | 171 |
| 11 | 6759 |

Inserción de datos

Ahora vamos a hacer operaciones de inserción de datos.

Nuevo cliente

Hay que insertar en la **tabla de clientes** los valores **gomez@gmail.com**, **913827832**. Realiza la misma consulta alterando el orden en el que aparecen los atributos en el esquema de la tabla.

```
INSERT INTO cliente VALUES('gomez@gmail.com', '913827832');
INSERT INTO cliente(telefono, correo_e)
VALUES('913827832', 'gomez@gmail.com');
```



Resultado

| Resultado | correo_e | telefono |
|-----------|----------------------|-----------|
| | alberca@hotmail.com | 912115339 |
| | albero@hotmail.com | 912115224 |
| | aliane@hotmail.com | 912115248 |
| | aliberti@hotmail.com | 912113502 |
| | alonso@hotmail.com | 912115214 |
| | alvarez@hotmail.com | 912115205 |
| | aparicio@hotmail.com | 912115309 |
| | araez@hotmail.com | 912113676 |
| | aragones@hotmail.com | 912115376 |
| | arias@hotmail.com | 912115373 |
| | gomez@gmail.com | 913827832 |

Nuevos datos de una reserva

Insertar la información correspondiente a la **reserva 10** para un viaje de ida y vuelta que parte de **Granada** el **29 de julio** con destino **Estambul**. El resto de los datos se desconocen.

```
INSERT INTO reserva(cod_reserva, origen, destino, fecha_sal, tipo)
VALUES(10, 'granada', 'Estambul', '29 julio', 'ida_vuelta');
```



Resultado

Modificación de datos

Incremento en un 5% de los precios de las reservas que incluyan dietas especiales

```
UPDATE reserva SET precio_total=precio_total*1.05 WHERE cod_reserva IN  
(SELECT cod_reserva FROM aparece WHERE comida_especial!='sin  
preferencia');
```

Para comprobarlo, vamos a mostrar los **códigos de reserva** junto con el tipo de **comida especial** que han solicitado los pasajeros de la misma y el **precio**, y veremos que las únicas reservas que no han modificado su valor son la **2** y la **7**, en las que ninguno de sus pasajeros ha solicitado una dieta especial:

```
SELECT r.cod_reserva, a.comida_especial, r.precio_total FROM reserva  
r, aparece a WHERE r.cod_reserva=a.cod_reserva;
```



Resultado

Resultado

| cod_reserva | comida_especial | precio |
|-------------|-----------------|---------|
| 5 | vegetariana | 711.9 |
| 6 | baja en caloría | 711.9 |
| 7 | sin preferencia | 345 |
| 8 | musulmana | 560.7 |
| 8 | sin lactosa | 560.7 |
| 1 | vegetariana | 879.27 |
| 9 | vegetariana | 245.7 |
| 9 | baja en caloría | 245.7 |
| 9 | sin preferencia | 245.7 |
| 10 | musulmana | (null) |
| 1 | baja en caloría | 879.27 |
| 2 | sin preferencia | 2345.5 |
| 3 | musulmana | 362.775 |
| 4 | sin lactosa | 2462.25 |

Incremento en un 5% de las reservas cuyo precio total sea inferior a la media de los precios calculados a partir de la descripción de las líneas

```
UPDATE reserva SET precio_total= precio_total*1.05 WHERE  
precio_total<= (SELECT AVG(precio_calculado) FROM (SELECT cod_reserva,  
SUM((tarifa+tarifa*tasas+gastos_gestion)*n_pasajeros) AS  
precio_calculado FROM lineas GROUP BY cod_reserva) AS temp);
```

En la tabla temporal **temp** calculo los **precios** según la información que aparece en las líneas (**subconsulta más interna**). A continuación calculo la **media** de éstos (**subconsulta externa**) y luego modifíco los precios de las **reservas** que sean **menores o iguales** al valor calculado. El precio medio calculado es: **AVG (precio_calculado): 16154.2727**. Se puede comprobar que todos los precios de las reservas se han modificado.



Resultado

Eliminación de datos

Dar de baja todos los vuelos de la empresa Kh

```
DELETE FROM vuelos WHERE empresa='kh';
```

Si mostramos el **código de vuelo** junto con la **empresa** que lo realiza de la tabla **vuelos** comprobaremos que ninguno de ellos pertenece a la empresa **Kh**;

```
SELECT cod_vuelo, empresa FROM vuelos;
```



Resultado

Resultado

| cod_reserva | n_tarjeta | tipo |
|-------------|-----------|----------|
| 1 | 4 | servired |
| 2 | 2 | visa |
| 3 | 1 | 4b |
| 4 | 7 | 6001 |
| 5 | 8 | servired |
| 6 | 5 | 4b |
| 7 | 5 | 4b |
| 8 | 10 | visa |

Resumen

En este tema hemos establecido una base de datos sobre la que realizar una serie de **consultas ejemplo** que te puedan ayudar a entender mejor el proceso de construcción de sentencias en el LMD de SQL. Se han establecido ejemplos que incluían desde **consultas simples** hasta **consultas complejas** y sentencias de **inserción, modificación y borrado** de filas de una tabla.



Para un mejor aprovechamiento de la lectura de la unidad se le ha proporcionado al alumno un fichero Excel que contiene los datos de todas las tablas de la base de datos original para que éste se encuentre guiado en todo el proceso.