Capítulo 3
La manipulación de los datos (LMD)

1. Introducción

El lenguaje de manipulación de datos permite a los usuarios y a los desarrolladores acceder a los datos de la base, modificar su contenido, insertar o eliminar filas.

Se basa en cuatro comandos básicos que son SELECT, INSERT, DELETE y UPDATE.

El administrador de la base de datos, que es el único que puede asignar o no los derechos, no siempre autoriza estos cuatro comandos.

Para un usuario x, se podrá indicar que solo puede utilizar el comando SELECT. Por razones evidentes de seguridad, no todos los usuarios tendrán acceso a los comandos de modificación.

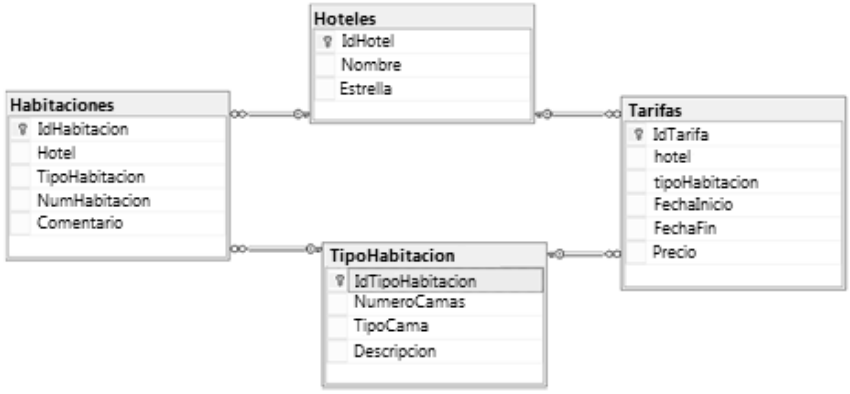
2. La selección de datos

El comando SELECT permite realizar consultas simples de forma rápida sin conocimientos profundos sobre lenguajes de programación. Es la cláusula básica la que permite indicar al servidor que queremos extraer datos.

De todos modos, puede ser muy potente si se conocen todas las funciones y todas las posibilidades del lenguaje. Se pueden realizar consultas complejas, con numerosas tablas pero siempre hay que poner atención al rendimiento que puede disminuir rápidamente en un comando SQL mal construido o que no utilice los índices correctos en las tablas. Hay que vigilar y utilizar las herramientas de análisis de consultas (vea el capítulo Profundizando - Algunos conceptos de rendimiento) antes de ejecutar una consulta sobre una base de datos real con tablas importantes.

Los principales elementos de una consulta de selección	
Cláusula	**Expresión**
SELECT	Lista columna(s) y o elementos de extracción
FROM	Tabla(s) fuente(s)
WHERE	Condición (condiciones) o restricción (restricciones), opcional
GROUP BY	Agrupación (agrupaciones), opcional
HAVING	Condición (condiciones) o restricción (restricciones) sobre la(s) agrupación (agrupaciones), opcional
ORDER BY	Ordenación (ordenaciones)

Las tablas de base que se utilizan en las siguientes secciones son:



2.1 El comando de selección de datos SELECT

El SELECT es el comando más importante y el más utilizado en SQL. Con este comando podemos recuperar las filas de una o más tablas y transformar los datos para su utilización o incluso realizar cálculos.

Vamos a describir poco a poco las posibilidades de este comando en los siguientes párrafos.

La utilización más normal consiste en seleccionar filas de una tabla:

```
SELECT NumeroCamas, Descripcion FROM TiposHabitacion;
```

En este ejemplo, hemos seleccionado dos columnas de la tabla TipoHabitacion.

El comando nos va a devolver todas las filas de la tabla para estas dos columnas.

Si se hubieran querido todas las columnas y todas las filas de la tabla, el comando habría sido este:

```
SELECT * FROM TiposHabitacion;
```

idTipo Habitacion	NumeroCamas	TipoCama	Descripcion
1	1	cama individual	1 cama individual con ducha
2	2	cama individual	2 camas individuales con ducha
3	2	cama individual	2 camas individuales con ducha y WC separados
4	1	cama doble	1 cama doble con ducha
5	1	cama doble	1 cama doble con ducha y WC separados
6	1	cama doble	1 cama doble con baño y WC separados
7	1	cama XL	1 cama doble grande con baño y WC separados

El asterisco es práctico cuando no se conocen los nombres de las columnas, pero el resultado es raramente legible con tablas que contengan un gran número de columnas. Para saber el número de columnas, haga antes un DESC de la tabla (DESC <nombre tabla>).

La sintaxis es simple:

```
SELECT <columna 1>, <columna 2> ... | * FROM <tabla1>, <tabla2> ...
```

Si algunas columnas tienen el mismo nombre perteneciendo a tablas distintas, habrá que añadir el nombre de la tabla delante de la columna para que el sistema sepa qué columna debe utilizar.

No es obligatorio poner el nombre de las tablas delante de cada columna, pero por una cuestión de legibilidad y de mantenimiento es preferible ponerlas cuando se trata de selecciones complejas.

```
SELECT Hoteles.Nombre
, Hoteles.Estrella
, Habitaciones.NumHabitacion
, TipoHabitacion.Descripcion
FROM Habitaciones, Hoteles, TipoHabitacion;
```

En la sección La utilización de los alias de este capítulo, veremos que para hacer la lectura más fácil podemos dar un alias a cada tabla. Este alias suele ser simple y permite encontrar fácilmente la tabla. Por ejemplo, HB para Habitaciones o TIPO para TiposHabitaciones.

2.2 Las opciones DISTINCT y ALL

Por defecto, en la ejecución de un SELECT se recuperan todas las filas (la opción ALL es automática). Si se quiere suprimir los duplicados hay que añadir la cláusula DISTINCT.

La cláusula DISTINCT se aplica a todas las columnas presentes en el comando.

Ejemplo

```
SELECT NumeroCamas, TipoCama, Descripcion FROM TiposHabitacion;
```

y

```
SELECT DISTINCT NumeroCamas, TipoCama, Descripcion
FROM TiposHabitacion;
```

Los dos SELECT anteriores tienen el mismo resultado aunque haya un duplicado en las tres primeras líneas. Las dos primeras columnas son iguales pero no la tercera.

NumeroCamas	TipoCama	Descripcion
1	**cama doble**	1 cama doble con baño y WC separados
1	**cama doble**	1 cama doble con ducha
1	**cama doble**	1 cama doble con ducha y WC separados
1	cama individual	1 cama individual con ducha
1	cama XL	1 cama doble grande con baño y WC separados

Por el contrario, si reducimos la selección a dos columnas:

```
SELECT NumeroCamas, TipoCama FROM TiposHabitacion;
```

se obtiene:

NumeroCamas	TipoCama
1	cama individual
2	cama individual
2	cama individual
1	cama doble
1	cama doble

Si se añade la cláusula DISTINCT, una de las filas que contienen '2' y 'cama individual' se eliminará.

```
SELECT DISTINCT NumeroCamas, TipoCama FROM TiposHabitacion;
```

Observación

La cláusula DISTINCT no se puede utilizar con operadores de agrupamiento (ver GROUP BY). De hecho los operadores de tipo COUNT o SUM eliminan automáticamente los duplicados.

NumeroCamas	TipoCama
1	cama doble
1	cama individual
1	cama XL
2	cama individual

2.3 Las ordenaciones

Cuando recuperamos columnas de una o varias tablas con una sentencia SELECT, normalmente es interesante de obtener un resultado ordenado sobre determinadas columnas.

Para esto, se utilizará la cláusula ORDER BY al final de la consulta. Se puede ordenar sobre cualquier columna de una tabla; para ello, es necesario que las columnas formen parte de la selección, pero no es obligatorio que se muestren.

La cláusula solo se puede utilizar una vez en una consulta y siempre debe ser la última cláusula de la consulta.

El orden por defecto es ascendente, indicado por ASC (del más pequeño al más grande). Es posible indicar que se desea realizar la ordenación de manera descendente, especificándolo con DESC.

Sintaxis del ORDER BY

```
ORDER BY <columna 1> [ASC|DESC], <columna 2> [ASC|DESC]...
```

Ejemplo

Ordenación sobre la FechaInicio de la tabla Tarifas de manera ascendente y sobre el precio de la tabla de manera descendente.

```
SELECT hotel
, tipoHabitacion
, FechaInicio
, precio
FROM Tarifas
ORDER BY FechaInicio,precio DESC;
```

Hotel	tipoHabitacion	FechaInicio	precio
1	7	01/04/2021	103,49
4	7	01/04/2021	103,49
4	6	01/04/2021	91,99
1	6	01/04/2021	91,99
1	5	01/04/2021	80,49
4	3	01/04/2021	80,49

Comprobamos que el orden se hace sobre la columna precio del más grande al más pequeño (DESC).

También existe la posibilidad de indicar el orden de la columna en la SELECT en lugar de su nombre, con la condición de que la columna esté presente en la selección.

Ejemplo sin el nombre de las columnas

```
SELECT hotel
, tipoHabitacion
, FechaInicio
, precio
FROM Tarifas
ORDER BY 3, 4 DESC;
```

Es cierto que esta notación funciona y permite no tener que volver a introducir el nombre de las columnas, pero en realidad no ayuda a la legibilidad de la consulta. Con selecciones múltiples, el orden rápidamente se hace poco legible para aquel que no lo ha escrito.