MEDAC
Instituto Oficial de Formación Profesional

ACCESO A DATOS
TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Manejo de conectores III: Sentencias

06

ÍNDICE

OBJETIVOS

Aprender las principales sentencias de definición de datos.

Aprender las principales sentencias de manipulación de datos.

Aprender a realizar consultas básicas.

Entender el concepto de transacción.

/ 1. Introducción y contextualización práctica

A lo largo de este tema, veremos muchos puntos relacionados con el **lenguaje estandarizado de base de datos SQL.** Un sistema de datos, como bien sabemos, almacena información sensible, y como es lógico, es necesario que dicha información esté almacenada de forma ordenada y estructurada.

Para ello, el lenguaje **nos dota de una serie de comandos** que nos serán de gran ayuda para esta estructura que comentamos y, en definitiva, para el diseño de nuestra base de datos.

Es por ello que en la siguiente unidad, **aprenderemos las sentencias** principales de definición de datos, de manipulación de datos, y también estudiaremos algunos ejemplos de cómo obtener información de nuestra fuente de datos, así como realizar consultas.

Planteamiento del caso práctico inicial

A continuación, vamos a plantear un caso a través del cual podremos aproximarnos de forma práctica a la teoría de este tema.

Escucha el siguiente audio donde planteamos la contextualización práctica de este tema, encontrarás su resolución en el apartado Resumen y resolución del caso práctico.

Fig. 1. Conectores..

Audio intro. "Consulta de datos"
https://bit.ly/39isXk4

/ 2. Ejecución de sentencias de definición de datos I: Creación de base de datos

En las siguientes diapositivas, mostraremos un primer acercamiento a las diferentes sentencias de definición de datos que disponemos para SQL. Tomaremos como ejemplo una base de datos MySQL.

Fig. 2. Almacenamiento tablas.

- **ALTER DATABASE:** este comando nos facilita cambiar características generales de una base de datos. Estas características son almacenadas en un archivo que suele ser "db.opt". Para usar este comando se necesita el permiso ALTER.

- **ALTER TABLE:** con este comando podremos modificar la estructura de una tabla que previamente hemos creado. Se pueden realizar numerosas acciones como agregar columnas, borrarlas, crear índices, borrarlos, cambiar la tipología de ciertas columnas, renombrar las columnas, también modificar la descripción de la tabla y la tipología de la misma.

- **CREATE DATABASE:** con este comando realizaremos la creación de una nueva base de datos, para ello es necesario dispones del permiso CREATE. Con "create_specification" se establecerán las distintas características de la base de datos. También están almacenadas en el fichero con extensión .opt en la raíz de la base de datos. Con la cláusula "character set", se especificará el set de caracteres por defecto de dicha base de datos que se está creando.

- **CREATE INDEX:** realmente, en muchas de las bases de datos, este comando se traduce a un comando ALTER TABLE para la creación de índices. Normalmente, con la creación de la tabla, se agregan todos los índices, pero con este comando podemos agregarlos manualmente después de haber creado una tabla. Para el tipo de columnas numeradas (columna1, columna2, etc.), se crea un índice de columnas múltiples. Los índices se forman al unir los valores de las columnas.

Audio 1. "Definición de datos"
https://bit.ly/30F67ze

/ 3. Ejecución de sentencias de definición de datos II: Creación y eliminación de tabla

Continuando con las sentencias de definición de datos que disponemos para SQL, podemos destacar las siguientes:

- **CREATE TABLE:** con este comando ejecutaremos la creación de una tabla con el nombre definido. Es evidente que también necesitaremos el permiso CREATE para la tabla. Existen algunas restricciones a la hora de establecer un nombre a la tabla definida. Ocurrirá un error si la tabla que estamos creando existe previamente en la base de datos que estamos trabajando. Se puede usar "TEMPORARY" y será solo visible mientras tengamos activa la conexión con la que estamos trabajando, después, dicha tabla no existirá. También podremos usar las claves "IF NOT EXISTS" para asegurarnos de que cierta tabla no existe.

- **DROP DATABASE:** usaremos este comando para realizar un borrado permanente de todas las tablas de nuestra base de datos y borrar, así, dicha base de datos. Evidentemente, es un comando muy peligroso, es por ello por lo que tendremos que tener específicamente habilitado el permiso de DROP. Si la base de datos que estamos borrando está enlazada simbólicamente, se borrarán ambos objetos.

- **DROP INDEX:** usando este comando y añadiendo el nombre del índice y la tabla especificada, se ejecutará un ALTER TABLE justo para borrar el índice que estamos indicando.

- **DROP TABLE:** con este comando podremos borrar una o más tablas en nuestra base de datos. El permiso DROP debe estar habilitado para nuestro usuario de la base de datos. Es otro comando que deberá de ser ejecutado con mucha precaución, ya que toda la información de la tabla que estamos indicando será eliminada. También podremos usar la clave "IF EXISTS" para evitar el error de cuando la tabla no exista. Este comando DROP TABLE realizará *commit* automáticamente al ser ejecutado.

- **RENAME TABLE:** con este comando podremos renombrar una o más tablas. Una vez se está ejecutando el renombrado, la tabla quedará temporalmente bloqueada hasta finalizar la transacción.

Fig. 3. Database concept.

/ 4. Ejecución de sentencias de manipulación de datos I: Inserción

A continuación, estudiaremos las distintas sentencias SQL que usamos para realizar manipulación de datos, también veremos la sintaxis de los comandos más importantes:

- **DELETE:** la sintaxis del comando DELETE:

```
DELETE [LOW_PRIORITY] [QUICK] [IGNORE] FROM tbl_name
[WHERE where_definition]
[ORDER BY ...]
[LIMIT row count]
```

Fig. 4. Sintaxis DELETE.

Con este comando eliminamos las filas de "tbl_name" que validan la condición expresada en "where_definition". Este comando nos devolverá el número de registros eliminados. Si no establecemos clausula "where", se eliminarán todas las filas de la tabla.

- **DO:** la sintaxis del comando DO:

```
DO expr [, expr] ...
```

Fig. 5. Sintaxis DO.

Con este comando ejecutaremos expresiones sin obtener ningún resultado. Realmente, es una forma de realizar SELECT (expresión), pero con la ventaja de que es más rápido cuando no es de interés el resultado.

- **HANDLER:** con este comando accederemos directamente a las distintas interfaces del motor de la tabla, tendremos comandos complementarios como OPEN, READ o CLOSE para leer ciertos datos de dicha tabla.

- **INSERT:** su sintaxis:

```
INSERT [LOW_PRIORITY | DELAYED | HIGH_PRIORITY] [IGNORE]
[INTO] tbl_name [(col_name,...)]
VALUES ({expr | DEFAULT},...),(...),...
[ ON DUPLICATE KEY UPDATE col_name=expr, ... ]
```

Fig. 6. Sintaxis INSERT.

Usaremos el comando INSERT para agregar nuevos registros en una tabla previamente definida. En relación a la forma INSERT–SET o INSERT–VALUES, se insertarán valores basados en las columnas de la tabla. También podremos encontrar INSERT–SELECT cogiendo registros de otra/s tabla/s.

- **LOAD DATA INFILE:** con este comando, podremos leer los registros.

Vídeo 1. "Inserts"
https://bit.ly/2BsVF5m

/ 5. Caso práctico 1: "Insertando datos"

Planteamiento: Disponemos en una base de datos una tabla llamada "CUSTOMER". Dicha tabla ha sido definida con los siguientes campos:

First_Name, *Last_Name*, *Age, Address*, *City, Country*, *Birth_Date*.

Se requiere introducir dos nuevas filas cuya edad sea 20 años para ambas, distinto nombre y apellidos, y pertenezcan a la ciudad de Málaga y Granada, respectivamente.

Nudo: ¿Cómo lo plantearías? Teniendo en cuenta las diferentes herramientas de las que disponemos en el lenguaje SQL, elegimos una sentencia de manipulación de datos. Usaremos INSERT.

Desenlace: Utilizaremos la sentencia INSERT para realizar el agregado de ambas filas, la sintaxis será la siguiente:

```
INSERT INTO CUSTOMER (First_Name, Last_Name, Age, Address, City, Country, Birth_Date )
VALUES ('juan', 'Lopez',20, 'avd jaen', Malaga, 'Spain', '1980-06-04');
```

Código 1. Sql 1.

```
INSERT INTO CUSTOMER (First_Name, Last_Name, Age, Address, City, Country, Birth_Date )
VALUES ('juan', 'Lopez',20, 'avd jaen', Granada, 'Spain', '1980-06-04');
```

Código 2. Sql 2.

Simplemente, destacar que realizamos la sentencia INSERT INTO, añadimos, después, el nombre de la tabla, y entre paréntesis, indicamos el nombre de los campos de la tabla donde vamos a hacer el INSERT. El orden es muy importante.

Después, directamente con la clave VALUES, introduciremos los valores teniendo en cuenta dos cosas:

1. La primera, el orden que hemos establecido de los campos.

2. La segunda, el tipo de dato que estamos introduciendo. Podemos observar como First_name lo introducimos con comillas simples al ser de tipo VARCHAR, sin embargo, Age lo introducimos directamente porque es un objeto de tipo NUMBER.

/ 6. Ejecución de sentencias de manipulación de datos II: Edición

Continuamos estudiando las distintas sentencias SQL que usaremos para realizar manipulación de datos, destacando las siguientes:

- **REPLACE:** aplicaremos la siguiente sintaxis:

```
REPLACE [LOW_PRIORITY | DELAYED]
[INTO] tbl_name [(col_name,...)]
```

Fig. 7. Sintaxis REPLACE.

Este comando realiza la misma función que INSERT con una excepción: se sobrescribirán los registros para los índices de tipo PRIMARY KEY o UNIQUE teniendo en cuenta que el registro anterior se borra antes de agregar el nuevo registro. Solo tendrá sentido, evidentemente, si la tabla contiene este tipo de índices.

- **SELECT:** usamos este comando para realizar consultas de registros de una o más tablas. Podremos realizar consultas simples, complejas o subconsultas. En el siguiente punto, dedicaremos todo un punto a estudiar las consultas y subconsultas, veremos su sintaxis y más detalles. Debido a su extensión, lo analizaremos más adelante.

- **TRUNCATE:** aquí podemos ver la sintaxis de TRUNCATE:

```
TRUNCATE TABLE tbl_name
```

Fig. 8. Sintaxis TRUNCATE.

Usaremos este comando para eliminar completamente una tabla. Es equivalente a un DELETE, pero con ligeras diferencias. Dependiendo del motor de base de datos, en algunos se realiza un DELETE tal como lo hemos estudiado, se borran todos los registros de esa tabla; y para otros motores de base de datos, se elimina el objeto completo de la tabla y se vuelve a crear. Hay que tener en cuenta que este tipo de operaciones no son transaccionales (que estudiaremos más adelante), lo que significa que nos dará un error si dicha tabla está ocupada en ese momento.

- **UPDATE:** revisaremos la sintaxis de este comando:

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
SET col_name1=expr1 [, col_name2=expr2 ...]
[WHERE where_definition]
[ORDER BY ...]
[LIMIT row_count]
```

Fig. 9. Sintaxis UPDATE.

Este nos actualizará la información de las columnas que le indiquemos en la sección SET con información nueva. Con la cláusula WHERE indicaremos qué registros deben actualizarse, y si no existe esta cláusula, se modificarán todos.

/ 7. Ejecución de consultas I: SELECT

En esta parte de la unidad, profundizaremos en el tema de consultas a base de datos, que está íntimamente relacionado con el comando **SELECT** que vimos en el apartado anterior.

Una consulta, básicamente, no es más que realizar una pregunta a base de datos con una serie de criterios, y esta ejecutará una respuesta a dicha pregunta. Podremos consultar una tabla o más de una.

Existen distintas cláusulas vinculadas a **SELECT:** podemos encontrar cláusulas de tipo HAVING, también podemos encontrar ORDER BY, UNION... la sentencia SELECT se puede combinar de diversas formas.

Cuando realizamos una consulta con SELECT, lo que obtenemos es una tabla ficticia, una serie de resultados que se relacionan acorde a nuestros requisitos en la consulta. Esta tabla de la que hablamos, evidentemente, no persiste en disco ni nada por el estilo, se mantiene en memoria mientras la estamos usando.

A continuación, veremos algunos de los puntos principales de la sintaxis de la sentencia SELECT:

- **SELECT:** Junto a la palabra clave "SELECT", podremos añadir [ALL/DISTINC], también asterisco [*] seguido de un listado de columnas [columnas], incluso podremos realizar un alias con la palabra clave AS [nombre del alias]. Con esta sintaxis podremos seleccionar ciertas columnas que van a ser mostradas junto con el orden. Con ALL o DISTINCT indicaremos si queremos mostrar, por ejemplo, filas duplicadas, o si en el conjunto de resultados, incluiremos filas únicas, respectivamente.

- **FROM:** Usaremos la palabra clave "FROM" y a continuación el nombre de la tabla/s o vista/s. Hasta aquí, con los conceptos básicos que tenemos, podríamos ejecutar perfectamente una consulta básica pidiendo todas las filas con todos sus campos de una supuesta tabla "CUSTOMER":

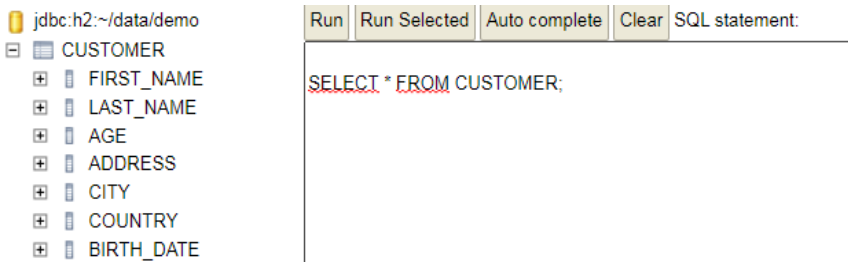

Fig. 10. Consulta customer.

- **WHERE:** Con dicha cláusula especificaremos algunas condiciones de filtro. Usaremos dicha cláusula cuando no queramos obtener el contenido total de la tabla o tablas que estemos consultando. A continuación, veremos algunas de las condiciones que podremos agregar a nuestras consultas.

/ 8. Ejecución de consultas: II: WHERE

En el apartado anterior, vimos la existencia de la cláusula WHERE. En este apartado, vamos a aprender a introducir algunas condiciones y expresiones lógicas en ella, de tal forma, que harán de filtro a la hora de preguntar a nuestra base de información y obtención de resultados. Algunas de las condiciones son las siguientes:

- **Operadores:** podemos encontrar mayor que (">"), mayor o igual que (">="), menor que ("<"), igual ("="), distinto que podemos expresarlo: ("<>") o también: ("!=").

- **Nulos:** podremos añadir IS NULL o IS NOT NULL si lo que queremos es comprobar que el valor de cierta columna sea nulo o no. Un valor es nulo cuando no existe valor, un registro en blanco no sería nulo.

- **LIKE:** usaremos esta clave cuando queramos realizar una comparación con los registros. Se usarán caracteres especiales. Estos caracteres son "_" y también "%". Con el "%" indicamos que puede ir cualquier cadena de caracteres en esa posición, y con el "_" sería el mismo concepto, pero, en este caso, solo con un carácter. Ejemplos:
  - **WHERE APELLIDO LIKE 'C%':** mostraremos resultados que coincidan donde el apellido empiece por "C".
  - **WHERE APELLIDO LIKE 'A_':** en este caso, el apellido empezaría por la letra "A" y luego le seguiría cualquier carácter.

Fig. 11. Filtrando consultas.

- **BETWEEN:** elegiremos BETWEEN cuando queramos establecer un rango de valores, podemos ver algún ejemplo:
  - **WHERE EDAD BETWEEN 3 AND 7:** mostraremos aquellas edades que estén entre 3 y 7, incluyendo dichos valores.
- **IN ():** usaremos la clave IN cuando nuestro objetivo sea mostrar una serie de resultados cuyos valores coincidan con los especificados en la clave. Ejemplo:
  - **WHERE EDAD IN (3, 4, 7, 8):** en este caso mostraremos, por ejemplo, los usuarios cuya edad coincida con las especificadas en la clave IN.

/ 9. Ejecución de consultas III: Ejemplos

Para terminar con la cláusula WHERE, comentaremos que todos estos operadores que hemos visto en el apartado anterior podremos combinarlos con operadores lógicos **OR (o), AND (y) y NOT (negativa)**, y ayudarnos de los paréntesis para establecer prioridades entre ellos.

Además, podemos destacar otra cláusula más, concretamente:

- **ORDER BY:** situaremos dicha cláusula después del WHERE, y como su nombre indica, la usaremos para establecer un orden a la hora de mostrar resultados según el campo o campos por los que queramos ordenar. Podremos ayudarnos de ASC o DESC si lo que deseamos es mostrar los resultados en orden ascendente en el caso ASC (de menor a mayor) o mostrarlos en orden descendente en el caso de DESC (de mayor a menor).

SELECT * FROM CUSTOMER;

FIRST_NAME	LAST_NAME	AGE	ADDRESS	CITY	COUNTRY	BIRTH_DATE
juan	Lopez	10	avd jaen	Badajoz	Spain	1980-06-04 00:00:00
pepe	Tejada	20	avd 1	Sevilla	Spain	1980-06-04 00:00:00
victor	Fernandez	22	avd 2	Jaen	Spain	1960-06-04 00:00:00
jose	Assensio	40	avd 3	Huesca	Spain	1930-06-04 00:00:00
Leo	Perez	30	avd 4	Orense	Spain	2000-06-04 00:00:00
Patricia	Aranda	11	avd hoho	Barceloona	Spain	1988-06-04 00:00:00
David	Lopez	105	avd peugeot	Madrid	Spain	1966-06-04 00:00:00
Enrique	Ferreras	70	avd jujer	Malaga	Spain	1980-06-04 00:00:00
Seluis	Guzman	45	avd Madrid	Valencia	Spain	1963-06-04 00:00:00

(9 rows, 8 ms)

Fig. 12. Todas las columnas.

Una vez visto lo más significativo entorno a las consultas de base de datos en relación al lenguaje SQL, a continuación, mostraremos algunos ejemplos de consultas. Trabajaremos con la base de datos de prueba que tenemos en nuestro entorno *Spring boot*, con la base de datos embebida H2, por ejemplo. Tenemos una tabla *"Customer"* con los campos o columnas de la Figura 12.

Básicamente, en esta consulta, estamos mostrando todas las columnas de la tabla "CUSTOMER". A continuación, mostraremos alguna consulta algo más compleja:

```
SELECT *
FROM CUSTOMER
WHERE (LAST_NAME LIKE 'L%') OR (LAST_NAME LIKE 'F%')
ORDER BY AGE DESC;
```

FIRST_NAME	LAST_NAME	AGE	ADDRESS	CITY	COUNTRY	BIRTH_DATE
David	Lopez	105	avd peugeot	Madrid	Spain	1966-06-04 00:00:00
Enrique	Ferreras	70	avd jujer	Malaga	Spain	1980-06-04 00:00:00
victor	Fernandez	22	avd 2	Jaen	Spain	1960-06-04 00:00:00
juan	Lopez	10	avd jaen	Badajoz	Spain	1980-06-04 00:00:00

(4 rows, 7 ms)

Fig. 13. Consulta campos.

Hemos realizado una consulta seleccionando todos los campos de la tabla CUSTOMER, filtrando por aquellos cuyo campo "LAST_NAME" empiece por "L" o por "F" y, además, ordenados de mayor a menor por el campo "AGE".

Vídeo 2. "Consultas"
https://bit.ly/3ePhiuh

/ 10. Gestión de transacciones

Para comenzar, diremos que una transacción en SQL son unidades o conjuntos de acciones que se realizan en serie y de forma ordenada en el sistema gestor de base de datos.

Los objetivos de las transacciones son dos:

- Proporcionar **consistencia** en la base de datos realizando secuencias de alta fiabilidad, de tal forma que se pueda volver a estados anteriores fácilmente.
- Ofrecer **aislamiento** cuando más de un aplicativo está accediendo a los datos simultáneamente.

Tendremos en cuenta los comandos de control que se realizan para la ejecución de transacciones en SQL:

- **Commit:** con este comando se persistirán los cambios en base de datos.
- **Rollback:** desharemos los cambios que se hubieran ejecutado hasta el momento y se abandonará la transacción.
- **Savepoint:** puntos donde se podrá almacenar y, en caso de rollback, se podrá volver a dicho punto de control.

10.1. La interfaz Statement

En la unidad didáctica anterior, estuvimos en contacto con esta interfaz cuando se trataba de realizar una conexión con un driver de una base de datos. Ahora más que nunca tiene sentido mencionar dicha interfaz, ya que es la encargada de ejecutar dichas sentencias en nuestro aplicativo y recoger los resultados para manipularlos más tarde.

Una vez se crea el objeto *Statement*, disponemos de un lugar adecuado para realizar consultas SQL. Podremos usar diferentes métodos para ello:

- **executeQuery (String):** utilizamos este método para realizar sentencias SELECT, siendo consultas que como resultado, este método nos devolverá un objeto ResulSet con toda la información resultante.

- **executeUpdate (String):** este método lo usaremos para realizar sentencias de manipulación de datos, ya sean INSERT, DELETE, UPDATE, etc. Una vez ejecutada la sentencia que le indiquemos, como String, nos devolverá un entero que contiene la cantidad de filas que han sido afectadas en la operación.

- **execute (String):** podemos usar este método para ejecutar cualquiera de las acciones propuestas en los dos casos anteriores. Simplemente, destacar que este método devolverá True si devuelve un *Resulset*, y para acceder a él, tendremos que ejecutar el método *getResulSet ()*, o false, si lo que estamos ejecutando, por ejemplo, es un UPDATE, en ese caso, si queremos saber las filas afectadas consultaríamos el método *getUpdateCount ()*.

/ 11. Caso práctico 2: "Lista con SELECT"

Planteamiento: A partir de los siguientes resultados tras realizar SELECT * FROM CUSTOMER:

SELECT * FROM CUSTOMER;

FIRST_NAME	LAST_NAME	AGE	ADDRESS	CITY	COUNTRY	BIRTH_DATE
juan	Lopez	10	avd jaen	Badajoz	Spain	1980-06-04 00:00:00
pepe	Tejada	20	avd 1	Sevilla	Spain	1980-06-04 00:00:00
victor	Fernandez	22	avd 2	Jaen	Spain	1960-06-04 00:00:00
jose	Assensio	40	avd 3	Huesca	Spain	1930-06-04 00:00:00
Leo	Perez	30	avd 4	Orense	Spain	2000-06-04 00:00:00
Patricia	Aranda	11	avd hoho	Barceloona	Spain	1988-06-04 00:00:00
David	Lopez	105	avd peugeot	Madrid	Spain	1966-06-04 00:00:00
Enrique	Ferreras	70	avd jujer	Malaga	Spain	1980-06-04 00:00:00
Seluis	Guzman	45	avd Madrid	Valencia	Spain	1963-06-04 00:00:00

(9 rows, 2 ms)

*Fig. 14. SELECT *.*

Se requiere obtener una lista de resultados de los clientes cuya edad sea mayor de los 40 y que esté ordenada por nombre de forma alfabética.

Nudo: Para realizar este tipo de consulta, usaremos la sentencia SELECT.

Desenlace: A continuación, mostraremos el resultado de la consulta:

```
SELECT * FROM CUSTOMER
WHERE AGE > 40
ORDER BY FIRST_NAME ASC;
```

FIRST_NAME	LAST_NAME	AGE	ADDRESS	CITY	COUNTRY	BIRTH_DATE
David	Lopez	105	avd peugeot	Madrid	Spain	1966-06-04 00:00:00
Enrique	Ferreras	70	avd jujer	Malaga	Spain	1980-06-04 00:00:00
Seluis	Guzman	45	avd Madrid	Valencia	Spain	1963-06-04 00:00:00

(3 rows, 4 ms)

Fig. 15. Select order by.

Como podemos comprobar, seleccionamos todas las columnas con * y nombramos la tabla CUSTOMER. En la cláusula WHERE, introducimos que la edad sea mayor de 40 con el operador ">", y por último, ordenaríamos en orden ascendente.

/ 12. Resumen y resolución del caso práctico de la unidad

En esta unidad, hemos tenido un contacto algo más directo con la capa del dato, de la información... Hemos podido entender cómo poder **obtener información preguntando a una fuente de datos.**

Hemos aprendido la diferencia entre una **sentencia de definición de datos** y una **sentencia de manipulación de datos.** Estudiamos sus principales tipos y características.

Avanzamos en el tema introduciéndonos en **el lenguaje SQL**, vimos sus cláusulas principales, de qué factores se compone, y aprendimos una consulta básica y algunos ejemplos. Por último, estudiamos el concepto de **transacción** y realizamos un pequeño recordatorio sobre la clase Statement para entender la aplicación de esta misma unidad.

```
select first_name, age, city
from customer
where (age BETWEEN 10 AND 40)
ORDER BY first_name desc;
```

FIRST_NAME	AGE	CITY
victor	22	Jaen
pepe	20	Sevilla
juan	10	Badajoz
jose	40	Huesca
Patricia	11	Barceloona
Leo	30	Orense

(6 rows, 2 ms)

Fig. 16. Consulta caso práctico.

Resolución del caso práctico inicial

A nuestro compañero Raúl le han encargado realizar una consulta de datos hacia una tabla llamada "CUSTOMER". Se prevé que va a necesitar una consulta cuyo resultado debe coincidir entre los clientes con edad entre 10 y 40 años, estando los resultados ordenados alfabéticamente en orden descendente, es decir, en orden inverso al alfabético. Además, va a necesitar seleccionar los campos: FIRST_NAME, AGE y CITY. La consulta sería, como podemos comprobar:

En primer lugar, seleccionamos los campos solicitados: "FIRST_NAME", "AGE" y "CITY", seleccionamos la tabla que queremos consultar, en este caso CUSTOMER. En la cláusula WHERE, usamos la clave BETWEEN para establecer un rango de edades, y por último, ordenamos por FIRST_NAME, pero de tipo DESC (descendente) contrario al orden alfabético.

/ 13. Webgrafía

https://docs.oracle.com/cd/E11882_01/server.112/e41084/toc.htm

https://manuales.guebs.com/mysql-5.0/sql-syntax.html#data-definition